DOE/NASA/0042-79/4
NASA CR 159608
BCS 40262-2

SIMWEST: A SIMULATION MODEL FOR WIND AND PHOTOVOLTAIC ENERGY STORAGE SYSTEMS (CDC Program Descriptions)

Volume II

A. W. Warren
R. W. Edsinger
J. D. Burroughs
Energy Technology Applications Division
Boeing Computer Services Company

August 1979

Prepared for
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
Lewis Research Center
Under Contract DEN3-42

for
U.S. DEPARTMENT OF ENERGY
Division of Energy Storage Systems

NOTICE

DOE/NASA/0042-79/4
NASA CR 159608
BCS 40262-2

SIMWEST: A SIMULATION MODEL FOR WIND AND PHOTOVOLTAIC ENERGY STORAGE SYSTEMS (CDC Program Descriptions)

Volume II

A. W. Warren
R. W. Edsinger
J. D. Burroughs
Energy Technology Applications Division
Boeing Computer Services Company
Seattle, Washington 98124

August 1979

Prepared for
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
Lewis Research Center
Cleveland, Ohio 44135
Under Contract DEN3-42

For
U.S. DEPARTMENT OF ENERGY
Division of Energy Storage Systems
Washington, D.C. 20545
Under Interagency Agreement EX-76-A-31-1026

FOREWORD

This report documents the CDC version of the SIMWEST computer programs developed by Boeing Computer Services Company under NASA Contract DEN3-42, "An Expanded System Simulation Model for Solar Energy Storage". The SIMWEST codes were originally developed for simulation of wind energy storage systems. The current version of these codes also includes solar-photovoltaic energy systems modeling. This project was conducted under the sponsorship of the Division of Energy Storage Systems, DOE, under the direction of Dr. G. C. Chang, and was administered by the NASA-Lewis Research Center Thermal and Mechanical Storage Section with Mr. L. H. Gordon and Mr. R. H. Beach as Project Managers.

This report is in two volumes.

I. CDC User's Manual
II. CDC Program Descriptions

The Boeing principal investigator for this project was Dr. A. W. Warren. Major contributors in the development of SIMWEST were Dr. R. W. Edsinger, Dr. J. D. Burroughs, and Dr. Y. K. Chan.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1.0 INTRODUCTION

This volume describes the computer programs for the CDC version of SIMWEST released in March 1979. Each of the following sections contain a verbal program description with macro flow charts, and source code listings for each major program entity. Section 2.0 describes the model generation precompiler program (EASY) which creates a Fortran model for the system to be simulated. Section 3.0 describes the simulation program (NONSIM). This is the executive program that exercises the Fortran model generated by the model generation program. Section 4.0 describes the file maintenance program (FILOAD). Section 5.0 describes the printer plotter program (NSMPPT) which is a post processor for the simulation program. The component library source listings are given in Section 7.0 of Volume I, CDC User's Manual.

2.0 MODEL GENERATION PROGRAM DESCRIPTION

2.1 INTRODUCTION

The Model Generation program accepts program commands which describe the system model in terms of standard components. Each standard component is represented by a subroutine. The program then constructs a FORTRAN model which consists of a series of calls to these subroutines. In addition to generating the FORTRAN source code for the system model, the Model Generation program produces a line printer drawn schematic diagram of the system and a list of the input data required to complete the model description.

Upon completion of model generation, the FORTRAN source code is compiled and the resultant object code is available as input to the simulation program. The model source code may be punched onto cards for storage or manipulation by the system analyst. The model object code is also stored on a permanent file. In this way a given model can be used for several simulation runs without having to regenerate the model for each analysis.

2.2 PROGRAM STRUCTURE

Figure 2.2-1 contains a macro flow diagram of the Model Generation program. This flow diagram shows the principle tasks of the program. For each task, a statement number in the main program is given along with the name of the principle subroutine that accomplishes the task.

The first task upon starting program execution is to obtain the current list of all standard components. The SIMWEST program was designed to be independent of the number or type of standard components. All that is required of the standard components is that their inputs, outputs, and table quantities be arranged according to certain rules discussed in Section 6.0, Volume I.

The sequence of performing the subsequent tasks is very model dependent. As each task is identified and performed, data describing the system model are

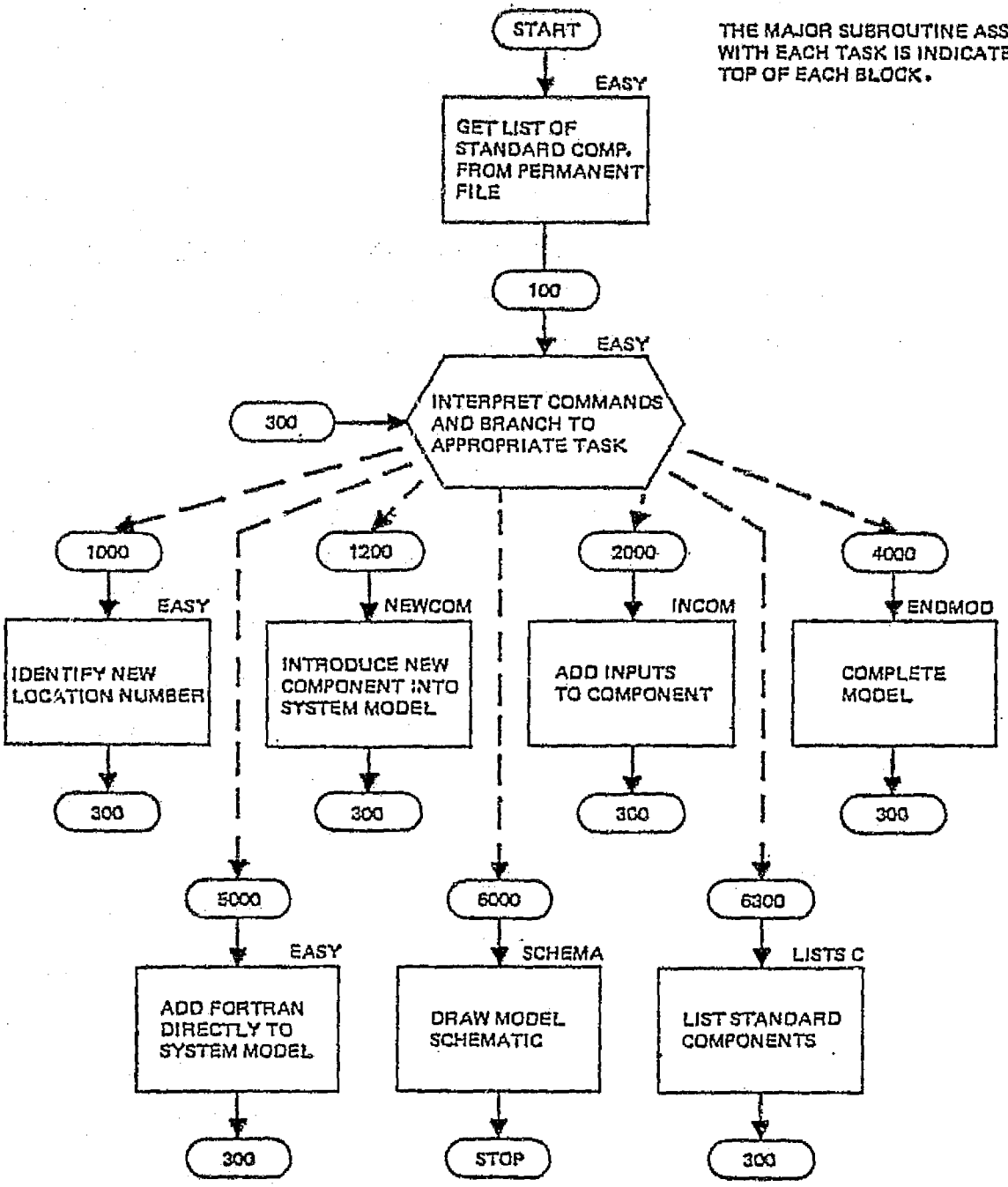

FIGURE 2.2-1 SIMWEST MODEL GENERATION PROGRAM - MACRO FLOW DIAGRAM

accumulated on a random access temporary file. This file, TAPE7, contains a list of inputs for each component in the system model. As inputs are satisfied by model connections, their names are modified to indicate the source of the input information. A list of model component names, CMPMOD, is kept in core. In addition to the component name, this list contains codes indicating the location of the component on the model schematic, the symbol to be used for the component and the number of inputs the component requires.

Once the END OF MODEL command is received, the data accumulated for the model is processed to generate the model source code and the model schematic diagram.

The following sections describe each of the major tasks shown in Figure 2.2-1. Source listings for all subroutines are included in Section 2.3.

2.2.1 Command Interpretation

The second task performed by the program is to begin the interpretation of data cards which contain the system model description commands. Figure 2.2-2 contains a macro flow diagram of the command interpretation process.

As each command card is read, it is printed to provide a record of progress through the model description. The model description is given as a series of "phrases". These phrases are identified in each card image by the routine, NXTPH, which locates one of the allowable phrase delimiters: comma, [,], equals, [=], left or right parenthesis, [()], or three or more blanks. When the end of a card is reached, a blank phrase is returned by NXTPH which causes a new command card to be read.

Each phrase is first tested against the set of command phrases, shown in Table 2.2-1. If a match is obtained between the first ten characters of the input phrase and one of the command phrases, the program branches to statement 400. At statement 400, tests are performed for unfinished tasks such as component definition that must be completed, or the end of the direct FORTRAN input task.

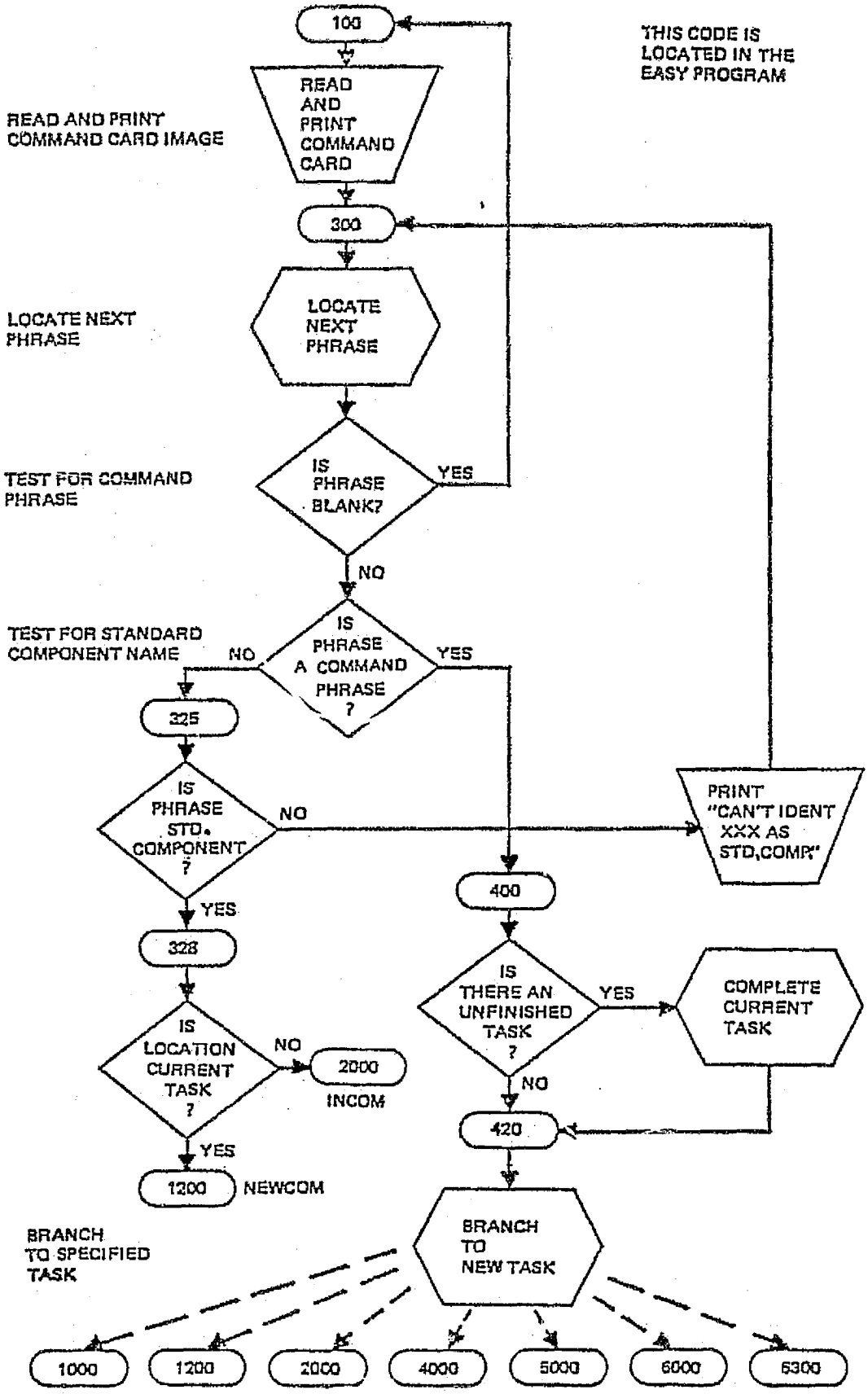

FIGURE 2.2-2 MODEL GENERATION COMMAND INTERPRETATION - MACRO FLOW DIAGRAM

TABLE 2.2-1 MODEL GENERATION PROGRAM COMMAND PHRASES

PHRASE	USE
ADD PARAMETERS	Direct addition of parameters to model
ADD STATES	Direct addition of states to model
ADD TABLES	Direct addition of tables to model
ADD VARIABLES	Direct addition of variables to model
DIAGNOSTIC CONTROL	Control diagnostic printout to model
END OF MODEL	Specify end of model description
FORTRAN STATEMENTS	Specify start of FORTRAN statements
INPUTS	Specify input components
LIST STANDARD COMPONENTS	Request listing of standard components
LOCATION	Specify component location on schematic
MODEL DESCRIPTION	Specify start of model description
PRINT	Requested printed model output
PUNCH	Request printed and punched model output

Once any unfinished task has been completed, a branch is made at statement 420 to the new task.

If the input phrase is not identified as a command phrase, its first two characters are compared to the list of standard component names, at statement 325. If the phrase is identified as a standard component, the program proceeds to either the new component routine, NEWCOM, or the component input routine, INCOM, depending on the current task.

If a particular command phrase requires additional modifying phrases, these phrases will be located on the command card and examined as to their suitability as a part of performing the requested task. For example, the INPUTS task will check for modifying port numbers or physical quantity names associated with the input component. The "suitability" of a phrase will be determined by assuring that it is numeric, a physical quantity name, etc. depending on the specified task.

2.2.2 LOCATION Command Execution

The LOCATION command introduces the definition of a new component into the system model. This command must be followed by a numeric phrase that specifies the component location on the model schematic diagram. Failure to furnish a numeric location number causes a warning to be printed and the component will not appear on the model schematic.

If the previous command involved the specification of a component LOCATION, or INPUTS, the input quantity list for that component is stored before examining the next phrase as a valid location number.

2.2.3 New Component Name Examination

The next phrase following the location number phrase should contain the name of a standard component. When this occurs, the subroutine NEWCOM is called.

If the name is not that of a standard component, a warning message will be printed and the program will continue on with command card interpretation.

A flow diagram of the NEWCOM subroutine is shown in Figure 2.2-3. The main purpose of the NEWCOM subroutine is to get copies of the input and output lists for the specified component. Master copies of these lists are stored on input file TAPE78 for all standard components. However, if a component has already appeared in the model description, an input list for that component will be stored on local file TAPE7. This copy of the input list must be used since it may contain information regarding previous connections.

Additional tasks performed by NEWCOM include storing the symbol number, location number, and number of inputs, in the component name. These three integer numbers are stored in the last six characters of the component's name by means of the PUTCOD routine. The PUTCOD routine allows up to 5 integer values to be stored in a single 10 character word. These integers may assume values between $\pm$ 2047. The routine GETCOD is used to retrieve these values. Figure 2.2-4 shows how the ten characters of each model component's name are used.

The PUTCOD routine is also used to store each model component's identification number, IDCOMP, in the LOCATION sequence array, SEQA. Components are assigned consecutive identification numbers as they first appear in a model description. These numbers define the sequence of component names in the model component name list, CMPMOD, and are used as the record numbers for the component input lists on the mass storage file TAPE7. The sequence array, SEQA, stores the component identification numbers in the sequence that is specified by the components' LOCATION statements. In some cases this sequence may differ from that of first appearance in the model description. The LOCATION statement sequence specifies the sequence that each model component subroutine is to be called in the system model.

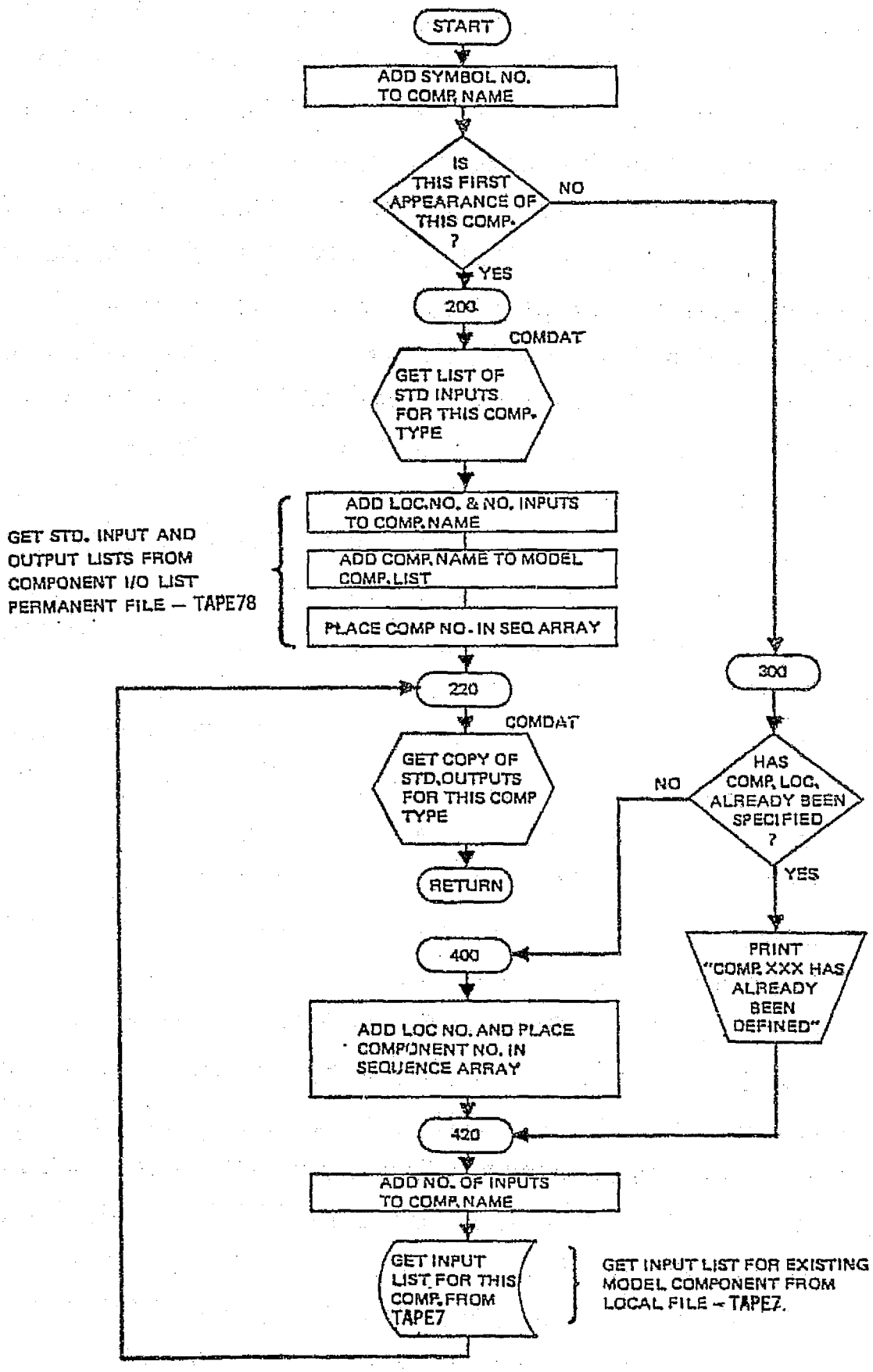

FIGURE 2.2-3 SUBROUTINE NEWCOM - MACRO FLOW DIAGRAM

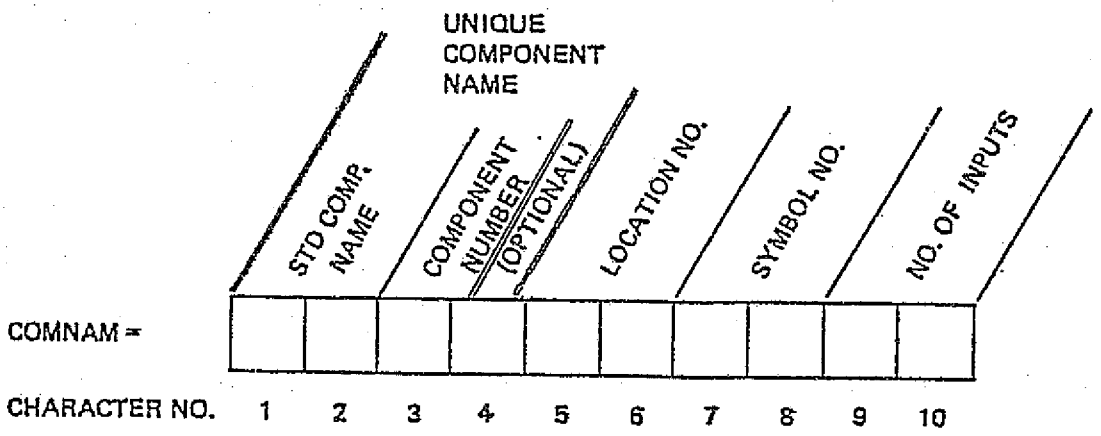

FIGURE 2.2-4 USE OF CHARACTERS IN COMPONENT NAMES

2.2.4 Inputs

The INPUTS command proceeds one or more instructions specifying those components which provide inputs to the component which has just been located. Component interconnections are made in the routine INCOM. Connections are recorded in the lists of inputs which are generated for each component as they are introduced into the model. The source of an input is indicated by replacing the standard physical quantity input name with the output quantity name of the source. Characters 4 through 7 of this name identifies the source component.

Figure 2.2-5 gives a macro-flow diagram of the INCOM routine. Upon entering the INCOM routine, input and output name lists are obtained for the upstream, i.e., input component. If this is the first appearance of this component, the input list is obtained from the input file TAPE78, via the routine COMDAT. If the component had previously appeared in the model, it will have an input list on local file TAPE7, which will be used. The next phrase after the upstream component name is then examined. There are three valid possibilities for this phrase. It can be blank or another standard component name in which case the default option of connecting all matching physical quantities at a pair of ports is taken. If this phrase is numeric, it is assumed that ports are being specified and all matching quantities at those ports are connected, via the routine PORTCN. If the phrase is alphanumeric and matches an output quantity of the upstream component, only the specified physical quantities are connected. Before returning from the INCOM routine, the input list for the upstream component is stored on TAPE7.

2.2.5 END OF MODEL Command Execution

The END OF MODEL command indicates the end of the model description. This command initiates the model generation process by the ENDMOD subroutine. The ENDMOD subroutine generates the FORTRAN source code for the system model routines EQMO, DATAIN, and BLOCK DATA MODEL and forms the model input requirements

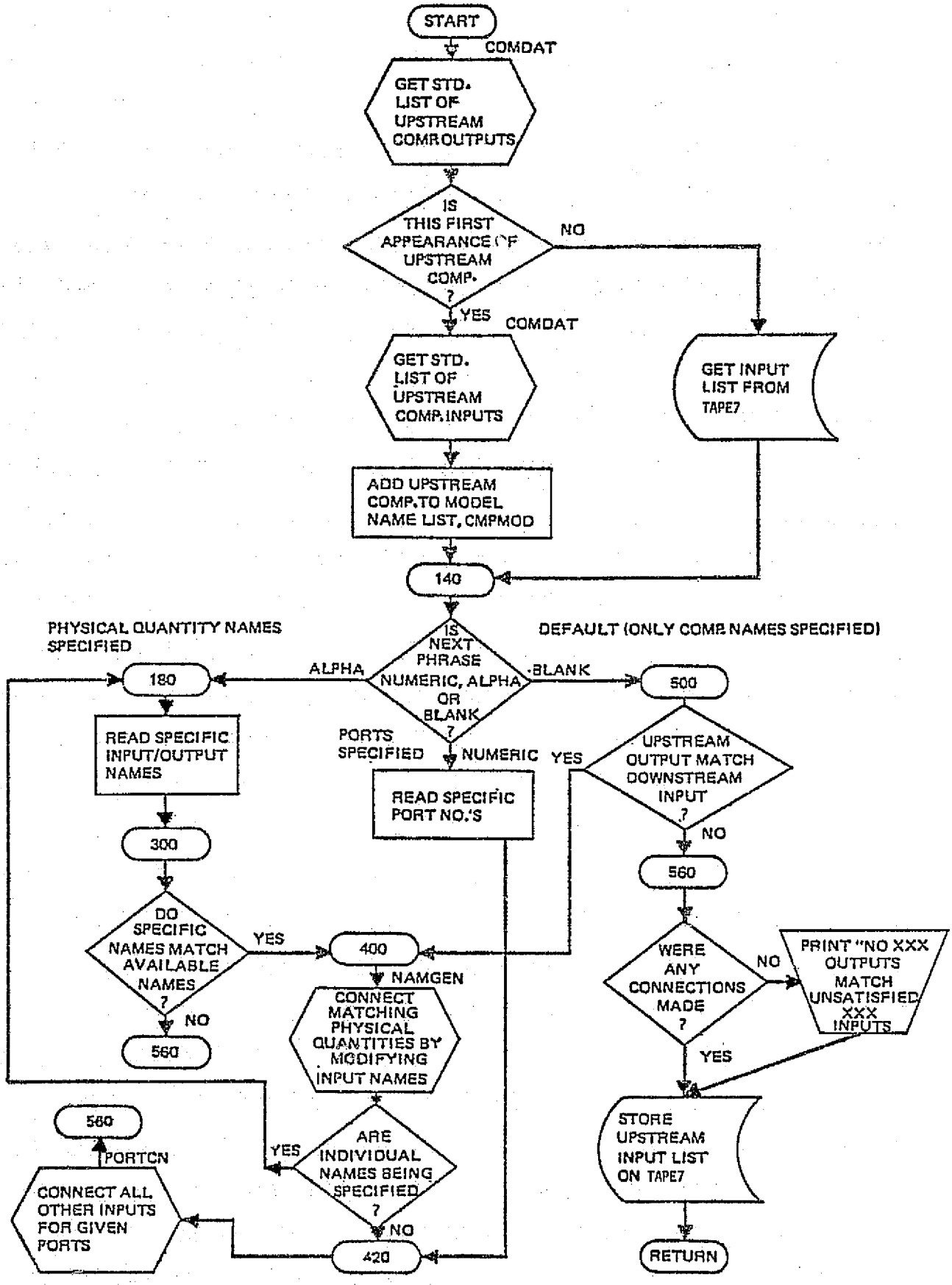

FIGURE 2.2-5 SUBROUTINE INCOM - MACRO FLOW DIAGRAM

list. The principle sources of data for the ENDMOD routine are: (1) the collection of input name lists for each model component, stored on TAPE7; (2) the list of model component names, CMPMOD; and (3) the location sequence of the model components, stored in SEQA. These lists describe all connections that have been made between standard components, the component names, and their location sequence in the model description. Figure 2.2-6 gives a macro flow diagram of the ENDMOD subroutine.

The subroutine COMORD is called to assure that the model equations are in explicit form. This is done by forming a binary connection matrix for the system model. Only those inter-component connections that involve variables are considered. For the model to be explicit, no component can receive an input from a variable which is calculated by a component occuring after it in the calculation sequence. Since states are not calculated by the component subroutines, (only their rates are), their values do not change within the EQMO routine and they can be used as inputs to any component in the model. If any direct FORTRAN STATEMENTS appear in the model, they are assumed to be dependent on all preceding components in the sequence. The COMORD routine calls the subroutine ORDER to test if the connection matrix is in lower triangular form, indicating an explicit system model. If this is not the case, ORDER will attempt to put the connection matrix in lower triangular form by rearranging the sequence of component subroutine calls. If a successful rearrangement can be made, a notice of those components whose sequence was changed will be printed. If the system model contains an implicit loop which cannot be removed by rearranging the component sequence, a warning will be printed, pointing out those components that form the implicit loop.

Once the sequence of model component calls has been established by COMORD, the source code for the subroutine calls is generated by the routines CALLCP and ENDCOM for standard components. This source code is temporarily stored on TAPE12. Lists of the state, variable, and parameter names contained in the model are also generated at this time and added to TAPE8, TAPE11, and TAPE10, respectively. These tasks for all system model components and any direct

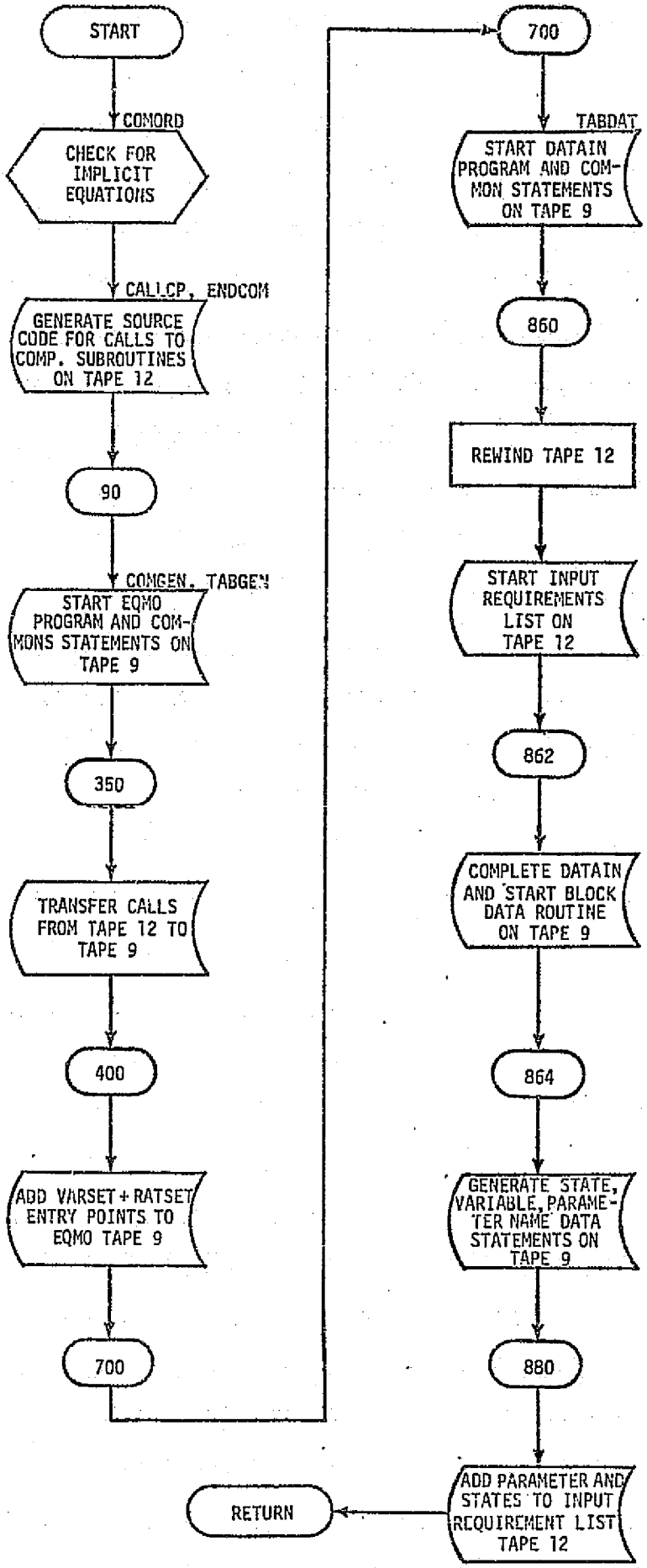

FIGURE 2.2-6 SUBROUTINE ENDMOD - MACRO FLOW DIAGRAM

FORTRAN STATEMENTS, are completed when statement number 90 of ENDMOD is reached.

The source code statements for EQMO are next written onto TAPE9. The subroutines COMGEN and TABGEN are used to generate common statements for the model states, variables, parameters, and tables. The calls to standard components are transferred from TAPE12 to TAPE9 and the VARSET and RATSET entry point statements are added to TAPE9 to complete the source code for EQMO.

At ENDMOD statement number 700, the generation of subroutine DATAIN begins. The statements in DATAIN provide default values for the integrator error controls and the value of .99999 for all model parameters. If tables are present in the models, the routine TABDAT generates the common /CTABLE/ containing the single array TABLES which is used to load tabular data into the model. TABDAT also loads the arrays, TABNAM, MAXDIM, and LOCTAB with the table names, maximum dimensions, and pointers that are used in the table data input process.

At ENDMOD statement number 860, TAPE12 is rewound and the start of the Input Requirement List for the model is placed on it. Subroutine TABCAL is called to place the table information in this list. A call to the table input routine TABIN is added to TAPE9, completing the DATAIN subroutine source code generation.

The BLOCK DATA MODEL routine source code is then added to TAPE9. The routine COMEQU is called once for each of the state, variable, and parameter name lists. This routine generates additional name arrays and equivalence statements whenever the number of names in a list exceeds 130. This is necessary to accommodate a compiler limitation of only 19 continuation cards in a single data statement. The NAMARY routine is used to transfer the state, variable, and parameter names from Tapes 8, 11, and 10 into source code data statements on TAPE9. The final task of the ENDMOD subroutine is to add the parameter and state names of the model to the Input Requirement List on TAPE12.

2.2.6 FORTRAN STATEMENTS Command Execution

The FORTRAN STATEMENTS command allows FORTRAN source statements to be inserted directly into the system model. When this command phrase is encountered, a component name of FORT is added to the model component name list. Subsequent lines of instructions are then placed on the source file, TAPE9. The first phrase of each subsequent line of instruction is compared with the SIMWEST command phrases. When a recognizable command is encountered, the direct FORTRAN mode terminates and the word FORT is written onto TAPE9 to mark the end of that block of FORTRAN statements. The recognized command is then executed.

Tests are included in the ENDMOD and COMORD routines to provide special handling of any "FORT" components. If the ENDMOD routine encounters a FORT component while generating calls to standard components, it transfers the FORTRAN source statements from TAPE9 to TAPE12, thus placing them in the proper sequence in the model equation subroutine, EQMO. If the COMORD routine encounters a FORT component while generating the model connection matrix, it specifies that inputs are provided to the "FORT" component by all previous components in the model sequence. This assumption assures that any inputs available to the FORTRAN statements in the given model sequence will also be available if the model component sequence is rearranged to remove implicit expressions.

2.3 MODEL GENERATION SOURCE LISTINGS

Compilation listings of the source code for the model generation program follows. Several subroutines such as NXTPH and KOMSTR are used in several of the programs and will be found in the source listings for the FILOAD program (Section 4.3). The names of the model generation routines, listed in alphabetical order, are as follows:

BLOCK DATA	LINE
CALLCP	LISTSC
COMEQU	NAMARY
COMGEN	NAMGEN
COMORD	NEWCOM
CONNCT	ORDER
EASY	PORTCN
ENDCOM	SCHEMA
ENDMOD	SYMBOL
HLINE	TABCAL
IJBIT	TABDAT
IJBIT1	TABGEN
INCOM	VLINE

```
 BLOCK DATA
 COMMON/COCINP/OCINPT(10)/COCOUT/OCOUTP(10)/COCCRI/OCCRIT(10)
 COMMON/COC/NOCIN,NOCOUT,NOC,NOCMOD,NOCCR,LOCOC,IOCAN,IXOC
 DATA OCINPT/100*(1H )/,OCOUTP/100*(1H )/
 DATA NOCIN/0/,NOCOUT/0/,NOC/-1/,NOCMOD/-1/,NOCCR/0/,LOCOC/-1/
1,IOCAN/0/,IXOC/1/
 END
```

```
CCALLCP
      SUBROUTINE CALLCP(COMNAM,NOCOMP,SOURCE,ISOUR,IVRSET,OUTPUT)
C  VERSION 2.                        REVISED: DEC 15 1975
C  PURPOSE:  TO INITIATE CALL GENERATION FOR STD. ECS COMPONENTS
C      CALL SEQUENCE: COMNAM - COMPONENT NAME
C                      NOCOMP - COMPONENT NUMBER
C                     SOURCE - SOURCE CODE ARRAY
C                     ISOUR  - SOURCE CODE ARRAY POINTER
C                     IVRSET - ARRAY CONTAINING VARSET,RATSET INFORMATIO
C                      OUTPUT - WORK ARRAY FOR OUTPUT   TABLE NAMES
      COMMON/CIO/IREAD,IWRITE,IDIAG/CTAB/NOTAB,TABNAM(1)
      COMMON /CORDER/NOX,NOV,NOP
      DIMENSION IVRSET(1),SOURCE(8)
     1,CALLS(2),OUTPUT(1),XDOT(2)
      DATA NEWCMP/10HNEW COMPNT/,COMMA/10H,         /
      DATA BLNK/10H           /
      DATA CALLS/20H      CALL   (      /,XDOT/20H,XDOT(    ),INT(    ) /
C --->      SAVE NO. OF VARIABLES AND STATES BEFORE COMPONENT IS FORMED
      I=4*NOCOMP-3
      CALL PUTCOD(I,IVRSET,NOV)
      I=4*NOCOMP-1
      CALL PUTCOD(I,IVRSET,NOX)
      WRITE(12,71)COMNAM
71    FORMAT(*C*/*C*,20X,*COMPONENT  *,A4/*C*)
C --->      LOAD SOURCE WITH CALL XX(
      DO 100 I=1,8
      IF(I.LE.2) GO TO 80
      SOURCE(I)=BLNK
      GO TO 100
80    SOURCE(I)=CALLS(I)
100   CONTINUE
C --->      LOAD STANDARD COMPONENT SUBROUTINE NAME
      CALL STRMOV(COMNAM,1,2,SOURCE,12)
      ISOUR=15
C --->      GET LIST OF TABLES FOR COMPONENT
      CALL COMDAT(COMNAM,4HTABS,NTAB,OUTPUT)
C --->      TEST IF TABLES ARE REQUIRED BY SUBROUTINE
      IF(NTAB.LE.0) GO TO 300
C --->      ADD TABLE ARGUMENTS TO CALL SEQUENCE
      IF(IDIAG.GT.60)WRITE(IWRITE,101)(OUTPUT(I),I=1,NTAB)
101   FORMAT(* CALLCP-TABLES*/(1X,6A10))
C --->      SCAN REQUIRED TABLES
      DO 200 I=1,NTAB
C --->      CONSTRUCT TABLE NAME
      ANAME=OUTPUT(I)
      CALL STRMOV(COMNAM,1,4,ANAME,4)
C --->      ADD TABLE NAME TO TABLE LIST
      NOTAB=NOTAB+1
      TABNAM(NOTAB)=ANAME
      IF(I.GT.1) CALL LINE(0,SOURCE,ISOUR,COMMA,1,12)
      CALL LINE(0,SOURCE,ISOUR,ANAME,7,12)
200   CONTINUE
C --->      GET LIST OF OUTPUT QUANTITIES FOR COMPONENT
300   CALL COMDAT(COMNAM,4HOUTP,NOUT,OUTPUT)
      IF(IDIAG.GT.60)WRITE(IWRITE,303)(OUTPUT(I),I=1,NOUT)
303   FORMAT(* CALLCP-OUTPUTS*/(1X,6A10))
```

```
C --->       SCAN OUTPUT QUANTITIES
      DO 400 I=1,NOUT
C --->       CONSTRUCT OUTPUT QUANTITY SPECIFIC NAME
      CALL NAMGEN(OUTPUT(I),COMNAM,ANAME)
C --->       GET 10TH CHARACTER IN STD. NAME TO DETERMINE IF QUANTITY
C  IS A STATE OR A VARIABLE
      CALL GETT(OUTPUT(I),10,TYPE)
C --->       TEST FOR STATE OR VARIABLE
      IF(TYPE.NE.BLNK) GO TO 320
C --->       INCREMENT VARIABLE COUNTER
      NOV=NOV+1
      WRITE(11,305)ANAME
305   FORMAT(A10)
      GO TO 330
C --->       INCREMENT STATE COUNTER
320   NOX=NOX+1
      WRITE(8,305)ANAME
330   IF(NTAB.GT.0.OR.I.GT.1) CALL LINE(0,SOURCE,ISOUR,COMMA,1,12)
C --->       ADD OUTPUT NAME TO CALL SEQUENCE
      CALL LINE(0,SOURCE,ISOUR,ANAME,7,12)
      IF(TYPE.EQ.BLNK) GO TO 400
C --->       CONVERT CURRENT NO. OF STATE TO BCD
      ENCODE(3,340,NO)NOX
340   FORMAT(I3)
C --->       LOAD CURRENT STATE NO. AS RATE SUBSCRIPT
      CALL STRMOV(NO,1,3,XDOT,7)
C --->       LOAD CURENT STATE NO. AS INT SUBCRIPT
      CALL STRMOV(NO,1,3,XDOT,16)
      CALL LINE(0,SOURCE,ISOUR,XDOT,19,12)
400   CONTINUE
      IF(IDIAG.GE.50)WRITE(IWRITE,405)SOURCE
405   FORMAT(* CALLCP-SOURCE*/(1X,6A10))
C --->       SAVE NO. OF VARIABLES AND STATES AFTER COMPONENT IS FORMED
      I=4*NOCOMP-2
      CALL PUTCOD(I,IVRSET,NOV)
      I=4*NOCOMP
      CALL PUTCOD(I,IVRSET,NOX)
      IF(IDIAG.EQ.55)WRITE(IWRITE,401)(IVRSET(I),I=1,NOCOMP)
401   FORMAT(* CALLCP-IVRSET*/3(2X,O20))
      RETURN
      END
```

```
CCOMEQU
      SUBROUTINE COMEQU(NAME,N)
C  VERSION 1.0                          REVISED: AUG 28 1975
C  PURPOSE:  CREATE EQUIVALENT NAME ARRAYS TO ALLOW DATA STATEMENTS
C            TO LOAD NAME LISTS EXCEEDING 130 NAMES.
C  CALL SEQUENCE:  NAME - NAME OF ARRAY TO BE EXTENDED
C                  N    - NUMBER OF NAMES IN LIST
C  DESIGNED BY:  J.D. BURROUGHS                    AUG 1975
C --->      CALCULATE NO. OF EXTENSIONS REQUIRED
      NEXT=(N-1)/130
      IF(NEXT.LE.0)RETURN
C --->      ADD AN EQUIVALENCE STATEMENT FOR EACH EXTENSION REQD.
      DO 100 I=1,NEXT
      J=130*I+1
C ---       CALCULATE NO. OF WORDS IN EXTENSION
      K=N-J+1
      IF(K.GT.130)K=130
      WRITE(9,81)NAME,I,K,NAME,J,NAME,I
81    FORMAT(6X,*DIMENSION *,A5,I2,*(*,I3,*)*/
     1 6X,*EQUIVALENCE(*,A5,*(*,I5,*),*,A5,I2,*)*)
100   CONTINUE
      RETURN
      END
```

```
CCOMGEN
      SUBROUTINE COMGEN(N,CNAME,NUNIT,IUNIT)
C  VERSION 3.                              REVISED: JULY 12 1977
C  PURPOSE: GENERATE COMMON STATEMENT GIVEN NAMES OF VARIABLES
C           STORED IN THE COMMON
C  CALL SEQUENCE: N      - NO. OF VARIABLES IN COMMON
C                 CNAME  - COMMON NAME. (2 CHARACTERS)
C                 NUNIT  - FILE NO. CONTAINING NAMES
C                 IUNIT  - FILE NO. TO WHICH SOURCE CODE IS TO
C                          BE WRITTEN.
      DIMENSION SOURCE(8),RNAMES(300)
      DATA INTEG/6HIJKLMN/
      REWIND NUNIT
C ---   CALC. NO. OF EXTENSIONS TO COMMON STATEMENT REQ D
      INT=0
      NEXT=(N-1)/156+1
      DO 400 J=1,NEXT
C ---    COMMON EXTENSION COUNTER
      K=J-1
C ---    NUMBER OF NAMES PER EXTENSION
      NAMES=N-K*156
C ---    LIMIT NO. OF NAMES PER COMMON TO 156
      NAMES=MIN0(NAMES,156)
C ---    GENERATE COMMON STATEMENT
C --->      FORM COMMON NAME
      SOURCE(1)=10H      COMM
      SOURCE(2)=10HON /  /
      CALL STRMOV(CNAME,1,2,SOURCE,15)
      DO 100 I=3,8
100   SOURCE(I)=10H
      ISOUR=18
C --->      SCAN NAMES
      DO 200 I=1,NAMES
      READ(NUNIT,105)ANAME
105   FORMAT(8A10)
C ---    TEST FOR INTEGER NAMES
      IF(ISCAN(ANAME,1,1,INTEG,1,6,K).EQ.0)GO TO 110
      INT=INT+1
      RNAMES(INT)=ANAME
110   IF(I.GT.1) CALL LINE(0,SOURCE,ISOUR,1H,,1,IUNIT)
      CALL LINE(0,SOURCE,ISOUR,ANAME,7,IUNIT)
C ----------  TEST FOR DIMENSIONED QUANTITIES
200   CONTINUE
      WRITE(IUNIT,105)SOURCE
400   CONTINUE
C ---    TEST IF INTEGER NAMES OCCURED
      IF(INT.EQ.0)RETURN
      SOURCE(1)=10H      REAL
      DO 500 I=2,8
500   SOURCE(I)=10H
C ---    SCAN INTEGER NAMES
      ISOUR=12
      DO 600 I=1,INT
      IF(I.GT.1)CALL LINE(0,SOURCE,ISOUR,1H,,1,IUNIT)
      CALL LINE(0,SOURCE,ISOUR,RNAMES(I),7,IUNIT)
600   CONTINUE
      WRITE(IUNIT,105)SOURCE
      RETURN
      END
```

```
CCOMORD
      SUBROUTINE COMORD(CMPMOD,NOCOMP,INPUTS)
C  VERSION 3.                          REVISED: JAN 6 1977
C  PURPOSE:  ORDER COMPONENTS SO THAT MODEL EQUATIONS ARE EXPLICIT
C  CALL SEQUENCE:  CMPMOD - ARRAY CONTAINING NAMES OF MODEL COMPONENTS
C                  NOCOMP - NUMBER OF COMPONENTS IN MODEL
C                  INPUT  - INPUT NAME ARRAY WORK SPACE
C  DESIGNED BY: J.D. BURROUGHS                          JULY 1975
      COMMON/CSEQ/NSEQ,SEQA(1)/CIO/IREAD,IWRITE,IDIAG
      DIMENSION CMPMOD(1),INPUTS(1),CONARR(667),ISEQ(200),IW1(200)
     1 ,IW2(200),W1(200),W2(200)
      EQUIVALENCE(W1,IW1),(W2,IW2)
C --->      TEST IF ALL COMPONENTS HAVE SEQUENCE NUMBERS
      IF(NSEQ.GE.NOCOMP)GO TO 100
C ==========  ASSIGN SEQUENCE NOS. TO UNSEQUENCED COMPONENTS
C --->      SCAN ALL MODEL COMPONENTS
      DO 85 I=1,NOCOMP
C --->      SKIP FORTRAN COMPONENTS
      IF(CMPMOD(I).EQ.4HFORT)GO TO 85
C --->      GET LOCATION CODE
      CALL GETCOD(3,CMPMOD(I),LOC)
      IF(LOC.GT.0)GO TO 85
C --->      INCREMENT SEQUENCE NO. COUNT
      NSEQ=NSEQ+1
      CALL PUTCOD(NSEQ,SEQA,I)
85    CONTINUE
C ===========  ZERO CONNECTION ARRAY
100   NWORDS=MIN0(NOCOMP*NOCOMP/60+1,60)
      DO 120 I=1,NWORDS
120   CONARR(I)=0.
C ===========  FORM CONNECTION ARRAY
C --->      SCAN MODEL COMPONENTS IN CURRENT SEQUENCE
      COMP=10H
      DO 400 I=1,NSEQ
C --->      GET COMPONENT NUMBER
      CALL GETCOD(I,SEQA,ICOMP)
C --->      TEST FOR FORTRAN COMPONENTS
      IF(CMPMOD(ICOMP).EQ.4HFORT)GO TO 360
C --->      GET NUMBER OF INPUTS TO ITH COMPONENT
      CALL GETCOD(5,CMPMOD(ICOMP),NINPUT)
C --->      SKIP COMPONENTS WITH ZERO INPUTS
      IF(NINPUT.LE.0)GO TO 400
C ===========  GET INPUT LIST FOR ITH COMPONENT
      CALL READMS(7,INPUTS,NINPUT,ICOMP)
      COMPS=10H
C --->      SCAN INPUTS
      DO 300 K=1,NINPUT
C --->      TEST TO IGNORE STATE INPUTS
      IF(KOMSTR(INPUTS(K),10,1,1HS,1).EQ.0)GO TO 300
C --->      GET NAME OF COMPONENT PROVIDING INPUT
      CALL STRMOV(INPUTS(K),4,4,COMP,1)
C ---       TEST TO SKIP PARAMETERS
      IF(COMP.EQ.10H           )GO TO 300
C --->      TEST TO SKIP SEARCH FOR SEQUENTIAL INPUTS FROM SAME COMPONEN
      IF(COMP.EQ.COMPS)GO TO 300
      COMPS=COMP
```

```
C ============  SCAN COMPONENTS TO LOCATE SEQUENCE NO. OF INPUT
      DO 200 J=1,NSEQ
      CALL GETCOD(J,SEQA,JCOMP)
C --->       COMPARE EACH COMPONENT WITH INPUT COMPONENT
      IF(KOMSTR(COMP,1,4,CMPMOD(JCOMP),1).EQ.0)GO TO 280
200   CONTINUE
      WRITE(IWRITE,201)COMP,CMPMOD(ICOMP)
201   FORMAT(/5X,15H*** WARNING ***,5X,*CAN T IDENTIFY   *,A4,*   AS A
     1VALID INPUT COMPONENT TO   *,A4/)
      GO TO 300
C --->       SET I J BIT = 1
280   CALL IJBIT1(CONARR,I,J,NSEQ)
300   CONTINUE
      GO TO 400
C ============  FOR FORTRAN COMPONENTS - REQUIRE ALL PREVIOUS COMPONENTS
360   DO 380 J=1,I
      CALL IJBIT1(CONARR,I,J,NSEQ)
380   CONTINUE
400   CONTINUE
C ============  LOAD SEQUENCE VECTOR
      DO 420 I=1,NSEQ
420   ISEQ(I)=I
C ============  ORDER COMPONENTS
      CALL ORDER(NSEQ,ISEQ,CONARR,IW1,IW2,IERROR,IB,IE)
      IF(IERROR.NE.0)GO TO 600
C --->       TEST FOR SUCCESSFUL ORDERING
      NWORDS=NSEQ/5+1
C ============  SAVE COPY OF SEQUENCE ARRAY
      DO 500 I=1,NWORDS
      W1(I)=SEQA(I)
500   CONTINUE
C --->       SET REARRANGEMENT COUNTER
      IREARR=0
C --->       SCAN CONPONENTS
      DO 540 I=1,NSEQ
C --->       TEST IF SEQUENCE HAS BEEN MODIFIED
      IF(ISEQ(I).EQ.I)GO TO 540
C --->       INCREMENT REARRANGEMENT COUNTER
      IREARR=IREARR+1
C --->       GET COMPONENT NUMBER
      CALL GETCOD(ISEQ(I),W1,JCOMP)
C --->       SAVE COMPONENT NAMES OF THOSE COMPONENTS WHOSE SEQUENCE HAS
      W2(IREARR)=CMPMOD(JCOMP)
      CALL PUTCOD(I,SEQA,JCOMP)
540   CONTINUE
C --->       TEST IF REARRANGEMENT OCCURED
      IF(IREARR.LE.0)RETURN
      WRITE(IWRITE,551)(W2(I),I=1,IREARR)
551   FORMAT(/5X,14H*** NOTICE ***,5X,*THE SEQUENCE OF THE FOLLOWING COM
     1PONENTS HAS BEEN ALTERED TO FORM AN EXPLICIT MODEL*//20(2X,A4)//)
      RETURN
```

```
C ==========  SCAN COMPONENTS THAT CAUSED IMPLICIT LOOP
600    J=0
       DO 620 I=IB,IE
       CALL GETCOD(IW2(I),SEQA,JCOMP)
       J=J+1
C ---->      SAVE NAMES OF COMPONENTS IN IMPLICIT LOOP
       W1(J)=CMPMOD(JCOMP)
620    CONTINUE
       WRITE(IWRITE,621)(W1(I),I=1,J)
621    FORMAT(/5X,15H*** WARNING ***,5X,*THE FOLLOWING COMPONENTS FORM AN
      1 IMPLICIT LOOP.  */*20(2X,A4)//)
       RETURN
       END
```

```
CCONNCT
      SUBROUTINE CONNCT(PAGE,NPAGE,LOC,INPUTS,NOIN,COMTAB,NOCOMP)
C  VERSION 2.                          REVISED: DEC 15 1975
C  PURPOSE:  FORM CONNECTING LINE BETWEEN TWO SPECIFIED COMPONENT
C            SYMBOLS AND LABEL IPPUTS
C  CALL SEQUENCE:  PAGE   - 13X56 ARRAY CONTAINING HOLLORITH
C                           REPRESENTATION OF A PAGE
C                  NPAGE  - CURRENT PAGE NO.
C                   LOC    - LOCATION OF SYMBOL TO WHICH LINE IS
C                           TO BE DRAWN
C                  INPUTS - ARRAY OF INPUT QUANTITY NAMES
C                  NOIN   - NO. OF INPUT QUANTITY NAMES
C                  COMTAB - TABLE OF ALL COMPONENT NAMES AND
C                           THEIR LOCATIONS
C                  NOCOMP - NO. OF COMPONENTS
      COMMON/CIO/IREAD,IWRITE,IDIAG
      DIMENSION PAGE(13,56),INPUTS(1),COMTAB(1)
C --->      RECEIVING COMPONENT LOCATION LINE NO.
      LOCLIN=7*((LOC-1)/10)+4
C --->      RECEIVING COMPONENT LOCATION COL. NO.
      LOCCOL=(MOD(LOC-1,10)+1)*13-6
      IF(IDIAG.EQ.30)WRITE(IWRITE,13)(INPUTS(I),I=1,NOIN)
13    FORMAT(* CONNCT-INPUTS*/(1X,6A10))
C --->      SCAN COMPONENTS LIST TO LOCATE INPUT COMP.
      DO 100 I=1,NOCOMP
      IF(KOMSTR(INPUTS,4,4,COMTAB(I),1).EQ.0)GO TO 120
100   CONTINUE
      CALL STRMOV(INPUTS,4,4,INLIN,1)
      WRITE(IWRITE,101)INLIN,LOC
101   FORMAT(/5X,32H *** WARNING ***  CAN T LOCATE  ,A4,
     1 *  AS AN INPUT COMPONENT TO LOCATION *,I4)
C --->      RETURN IF INPUT COMPONENT ISN T IN COMTAB LIST
      GO TO 540
C --->      GET LOCATION OF INPUT COMPONENT
120   CALL GETCOD(3,COMTAB(I),ILOC)
C --->      DETERMINE PAGE OF INPUT COMPONENT
      IPAGE=(ILOC/100)*100
C --->      COMPARE INPUT COMP. PAGE TO CURRENT PAGE
      IF(IPAGE.NE.NPAGE)GO TO 420
C --->      CONVERT GENERAL PAGE LOC TO LOCAL PAGE LOC
      ILOC=ILOC-IPAGE
C --->      CALC. LOC. LINE AND COL. NO. FOR INPUT COMPONENT
      ILIN=7*((ILOC-1)/10)+4
      ICOL=(MOD(ILOC-1,10)+1)*13-6
C ---       TEST FOR INPUTS FROM DOWNSTREAM COMP.
      IDS=0
      IF(KOMSTR(INPUTS,8,1,1H ,1).NE.0)IDS=1
C --->     TEST IF RECEIVING COMP. AND INPUT COMP. ARE ON SAME LINE
      IF(ILIN-LOCLIN)200,130,220
C --->      SAME LINE.  TEST IF LEFT OR RIGHT
130   IF(ICOL.GE.LOCCOL)GO TO 140
C --->      SAME LINE AND INPUT IS TO LEFT
135   INCOL=ICOL+6
      IRCOL=LOCCOL-5
      ITC=IRCOL-7
      ITL=2-LOCLIN
      GO TO 160
```

```
C --->      SAME LINE AND INPUT IS TO RIGHT
140   INCOL=ICOL-5
      IRCOL=LOCCOL+6
      ITC=IRCOL+1
      ITL=LOCLIN+2
C --->      ADD HORIZONTAL LINE
160   IF(IDS.NE.0)GO TO 500
      CALL HLINE(PAGE,ILIN,INCOL,IRCOL)
      GO TO 500
C --->      INPUT IS ABOVE.  TEST IF LEFT OR RIGHT
200   IF(ICOL-LOCCOL)300,240,320
C --->      ABOVE AND SAME COLUMN
240   INLIN=ILIN+3
      IRLIN=LOCLIN-4
      ITC=LOCCOL+3
      ITL=1-IRLIN
      GO TO 280
C --->      INPUT IS BELOW.  TEST IF LEFT OR RIGHT
220   IF(ICOL-LOCCOL)340,260,360
C --->      BELOW AND SAME COLUMN
260   INLIN=ILIN-4
      IRLIN=LOCLIN+3
      ITC=LOCCOL-8
      ITL=IRLIN+1
C --->      ADD VERTICAL LINE
280   IF(IDS.NE.0)GO TO 500
         CALL VLINE(PAGE,ICOL,INLIN,IRLIN)
      GO TO 500
C --->      INPUT IS IN UPPER LEFT QUAD.
300   IF(IDS.NE.0)GO TO 135
      LIN=ILIN+1
      INCOL=ICOL+6
      IRCOL=LOCCOL-1
      ICO=IRCOL
      INLIN=LIN
      IRLIN=LOCLIN-4
      ITC=LOCCOL-9
      ITL=1-IRLIN
      GO TO 400
C --->      INPUT IS IN UPPER RIGHT QUAD.
320   IF(IDS.NE.0)GO TO 240
      LIN=LOCLIN-1
      INCOL=ICOL-1
      IRCOL=LOCCOL+6
      ICO=INCOL
      INLIN=ILIN+3
      IRLIN=LIN
      ITC=LOCCOL+7
      ITL=1-IRLIN
      GO TO 400
C --->      INPUT IS IN LOWER LEFT QUAD.
340   IF(IDS.NE.0)GO TO 260
      LIN=LOCLIN+1
      INCOL=ICOL+1
      IRCOL=LOCCOL-5
      ICO=INCOL
```

```
      INLIN=ILIN-4
      IRLIN=LIN
      ITC=IRCOL-6
      ITL=IRLIN+1
      GO TO 400
C --->      INPUT IS IN LOWER RIGHT QUAD.
360   IF(IDS.NE.0)GO TO 140
      LIN=ILIN-1
      INCOL=ICOL-5
      IRCOL=LOCCOL+1
      ICO=IRCOL
      INLIN=LIN
      IRLIN=LOCLIN+3
      ITC=IRCOL+2
      ITL=IRLIN+1
C --->      ADD VERTICAL LINE SEGMENT
400   CALL VLINE(PAGE,ICO,INLIN,IRLIN)
C --->      ADD HORIZONTAL LINE SEGMENT
      CALL HLINE(PAGE,LIN,INCOL,IRCOL)
      GO TO 500
C --->      INPUT IS FROM ANOTHER PAGE
C ---       TEST TO PREVENT OFF PAGE SYMBOL FROM FALLING OFF PAGE
420   IF(LOCLIN+7.GT.56.OR.LOCCOL-16.LT.1)GO TO 440
C --->      GENERATE EXTERNAL PAGE SYMBOL
      CALL PUTT(PAGE(1,LOCLIN+3),LOCCOL-5,1H/)
      CALL PUTT(PAGE(1,LOCLIN+4),LOCCOL-7,1H/)
      CALL STRMOV(7H******/,1,7,PAGE(1,LOCLIN+5),LOCCOL-15)
C --->      PLACE EXTERNAL PAGE NO. IN EXTERNAL PAGE SYMBOL
      IPAGE=IPAGE/100
      ENCODE(8,421,IPAGE)IPAGE
421   FORMAT(5H*PAGE,I2,1H*)
      CALL STRMOV(IPAGE,1,8,PAGE(1,LOCLIN+6),LOCCOL-16)
      CALL STRMOV(6H******,1,6,PAGE(1,LOCLIN+7),LOCCOL-15)
440   ITC=LOCCOL-16
      ITL=LOCLIN+8
C --->      ADD TEXT TO INPUT LINE
500   K=ISIGN(1,ITL)
      ITL=IABS(ITL)
      IF(NOIN.LT.1)GO TO 540
C ---       PREVENT LABELS FROM FALLING OFF SIDES OF PAGE
      IF(ITC.LT.1)ITC=1
      IF(ITC.GT.123)ITC=123
C ---       TEST FOR LABELS GOING OFF TOP OR BOTTOM OF PAGE
      IDS=ITL+K*(NOIN-1)
C ---       REVERSE DIRECTION OF COLUMN TO PREVENT LOSS OF LABELS
      IF(IDS.LT.1.OR.IDS.GT.56)K=-K
C --->      SCAN INPUTS FROM INPUT COMP.
      DO 520 I=1,NOIN
C ---       TEST TO ASSURE THAT LABELS STAY ON PAGE
      IF(ITL.LT.1.OR.ITL.GT.56)GO TO 540
C --->      ADD INPUT NAMES TO PAGE
      CALL STRMOV(INPUTS(I),1,7,PAGE(1,ITL),ITC)
C --->      INCREMENT PRINT LINE EITHER UP OR DOWN
      ITL=ITL+K
520   CONTINUE
540   NOIN=0
      RETURN
      END
```

```
CEASY
      PROGRAM EASY(INPUT=100,OUTPUT=100,TAPE5=INPUT,TAPE6=OUTPUT
     1 ,TAPE7=100,TAPE8=100,TAPE9=100,TAPE10=100,TAPE11=100,TAPE12=100,
     2 TAPE4=100,TAPE78=100,PUNCH=100,TAPE3=PUNCH)
C  VERSION 4.                                  REVISED JUNE 27 1977
C   PURPOSE:  TO GENERATE FORTRAN SOURCE OF ECS MODEL IN THE
C             FORM REQUIRED BY THE NONSIM PROGRAM.
C  LIMITATIONS:  ARRAY DIMENSIONS IMPOSE THE FOLLOWING LIMITS
C          LIMITED QUANTITY      CURRENT VALUE      ARRAYS IMPOSING THE LI
C
C       STANDARD COMPONENTS          K = 150     MSI(I)      K=(I-3)/6
C                                    K = 150     CMPNTS(I)   K=I-1
C
C     STD. COMPONENTS PER MODEL      K = 200     IVRSET(I)   K=I*5/4  (ENDMOD)
C                                    K = 200     CONARR(I)   K=(60*I)**.5 (COMORD
C                                    K = 200     ISEQ(I)     K=I    (COMORD)
C                                    K = 200     IW1(I)      K=I    (COMORD)
C                                    K = 200     IW2(I)      K=I    (COMORD)
C                                    K =  200    CMPMOD(I)   K=I
C                                    K = 200     ININDEX(I)  K=I-1
C                                    K = 200     SEQA(I)     K=5*I
C
C      INPUTS FOR ANY STD. COMP.     K =  63     DINPUT(I)   K=I-1
C                                    K =  63     UINPUT(I)   K=I-1   (INCOM)
C
C      OUTPUTS FOR ANY STD. COMP.    K =  63     DOUT(I)   K=I-1
C                                    K =  63     UOUT(I)     K=I-1   (INCOM)
C
C       TABLES PER STD. COMP.        K =  15     TABLE(I)    K=I    (FILOAD)
C
C       TABLES PER MODEL             K = 100     TABNAM(I)   K=I    (FILOAD)
C
C     OPTIMAL CONTROLLER INPUTS      K =  63     OCINPT(I)   K=I
C
C     OPTIMAL CONTROLLER OUTPUTS     K =  63     OCOUTP(I)   K=I
C
C     OPTIMAL CONTROLLER CRITERIA    K =  63     OCCRIT(I)   K=I
C
C    DESIGNED BY: J.D.BURROUGHS                          DATE: MAY 1974
      COMMON/CMSI/MSI(897)/CIO/IREAD,IWRITE,IDIAG/CORDER/NOX,NOV,NOP
     1/CTITLE/TITLE(7)/CSEQ/NSEQ,SEQA(40)/CTAB/NOTAB,TABNAM(100)
      COMMON/COCINP/OCINPT(63)/COCOUT/OCOUTP(63)/COCCRI/OCCRIT(63)
      COMMON/COC/NOCIN,NOCOUT,NOC,NOCMOD,NOCCR,LOCOC,IOCAN,IXOC,IUOC
      DIMENSION ICOM(8),CMMNDS(21),SOURCE(8),DINPUT(64),DOUT(64)
      DIMENSION CMPNTS(151),CMPMOD(200),ININDEX(201)
      DATA NOCOMP/0/,ITASK/6/,IPUNCH/0/
      DATA BLNK/10H          /,ICMMAX/21/
      DATA CMMNDS/210HLOCATION  INPUTS    FORTRAN STEND OF MODXXXXXXXXXX
     1MODEL DESCPRINT     XXXXXXXXXXPUNCH     DIAGNOSTICADD STATESADD VA
     2RIABADD PARAMEADD TABLESLIST STANDO.C. INPUTO.C. OUTPUO.C. ORDERO.
     3C. MODELO.C. CRITEO.C. ANALY/
      IDIAG=0
      IREAD=5
      IWRITE=6
      WRITE(IWRITE,11)
11    FORMAT(1H1,10X,*INPUT COMMANDS*/)
```

```
C --->      OPEN STANDARD COMPONENT FILE
      CALL OPENMS(78,MSI,897,1)
C --->      OPEN MODEL COMPONENT INPUT FILE
      CALL OPENMS(7,ININDEX,201,0)
C --->      OBTAIN STD. COMPONENT NAMES FROM PERMANENT FILE
      CALL READMS(78,ICPMAX,1,6HCMPNTS)
      CALL READMS(78,CMPNTS,ICPMAX,6HCMPNTS)
      DO 20 I=2,ICPMAX
20    CMPNTS(I-1)=CMPNTS(I)
      ICPMAX=ICPMAX-1
C --->      READ DATA CARD
100   READ(IREAD,101)ICOM
101   FORMAT(8A10)
      IF(EOF(IREAD))6260,200
200   WRITE(IWRITE,201)ICOM
201   FORMAT(/* COMMAND CARD ---> *,8A10)
C --->      DIAGNOSTIC PRINTS
      IF(IDIAG.EQ.10)WRITE(IWRITE,205)(CMMNDS(I),I=1,ICMMAX)
205   FORMAT(* COMMANDS*/10(1X,A10))
      IF(IDIAG.EQ.20)WRITE(IWRITE,207)(CMPNTS(I),CMPNTS(I),I=1,ICPMAX)
207   FORMAT(* STD. COMPONENTS*/4(1X,A10,1X,O20))
      ENDFILE IWRITE
C --->      INDEX FOR DATA CARD COLUMN
      INDEX=1
C --->      LOCATE NEXT PHRASE
300   CALL NXTPH(ICOM,INDEX,PHRS)
320   IF(PHRS.EQ.BLNK) GO TO 100
C --->      SEARCH COMMAND LIST
      CALL LCMPH(PHRS,CMMNDS,ICMMAX,1,NTASK)
C --->      NTASK = NEW TASK INDICATOR
      IF(NTASK.NE.0) GO TO 400
C --->      TEST FOR DIRECT MODEL MODES AND O.C. INPUTS
      GO TO(325,325,5000,325,325,325,325,325,325,325,
     1 5100,5200,5300,5400,325,7000,7000,7000,7000,7000,
     2 7000),ITASK
C --->      SEPERATE STANDARD COMPONENT NAME FROM SPECIFIC COMPONENT NAM
325   COMP=BLNK
      CALL STRMOV(PHRS,1,2,COMP,1)
C --->      SEARCH COMPONENT NAME LIST
      DO 326 ICOMP=1,ICPMAX
      IF(KOMSTR(CMPNTS(ICOMP),1,2,COMP,1).EQ.0)GO TO 328
326   CONTINUE
      ICOMP=0
      GO TO 330
328   IF(ITASK.EQ.1) GO TO 1200
      GO TO 2000
330   WRITE(IWRITE,335)COMP
335   FORMAT(/5X,34H *** WARNING ***  CAN T IDENTIFY  ,A10,*AS A STANDAR
     1D COMPONENT.*)
      IF(ITASK.EQ.2)GO TO 300
      ITASK=6
      NEWC=0
      GO TO 300
C --->      NEW COMMAND IDENTIFIED
400   LTASK=ITASK
      ITASK=NTASK
```

```
      IF(LTASK.EQ.3)WRITE(9,401)
401   FORMAT(*FORT*)
C --->      TESTS FOR UNFINISHED BUSINESS
      IF(LTASK.EQ.1.OR.LTASK.EQ.2) GO TO 410
C --->      BRANCH TO NEW TASK
420   GO TO(1000,2000,500,4000,4000,520,6000,100,5900,1400,
     1 300,300,300,300,6300,300,300,300,300,300,
     2 7100),ITASK
410   IF(ITASK.EQ.2) GO TO 300
      GO TO 3000
C ====================  FORTRAN STATEMENTS   ITASK = 3
500   NOCOMP=NOCOMP+1
C ----      ADD COMP. NO. TO COMPONENT SEQUENCE LIST
      NSEQ=NSEQ+1
      CALL PUTCOD(NSEQ,SEQA,NOCOMP)
      CMPMOD(NOCOMP)=4HFORT
      GO TO 100
C ====================  MODEL DESCRIPTION    ITASK = 6
520   NEWC=0
      NOV=0
      NOX=0
      NOP=0
      NOCOMP=0
      NSEQ=0
      NOTAB=0
      NOCIN=0
      NOCOUT=0
      NOC=-1
      NOCMOD=-1
      NOCCR=0
      LOCOC=-1
      IOCAN=0
      IXOC=1
      REWIND 8
      REWIND 10
      REWIND 11
C --->      LOAD TITLE
      DO 530 I=1,7
530   TITLE(I)=10H
      I=INDEX+1
      J=80-INDEX
      CALL STRMOV(ICOM,I,J,TITLE,1)
      GO TO 100
C --->      INITIATE NEW COMPONENT
C --->      GET COMPONENT LOCATION NUMBER
C ====================  LOCATION      ITASK = 1
1000  CALL NXTPH(ICOM,INDEX,LOCNO)
      CALL NUMERC(LOCNO),RETURNS(1100)
      GO TO 300
1100  WRITE(IWRITE,1101)LOCNO
1101  FORMAT(/5X,18H *** WARNING ***  ,A10,*  IS NOT A VALID LOCATION NU
     1MBER*)
      CALL STRMOV(LOCNO,1,10,PHRS,1)
      LOCNO=10H-100
      GO TO 320
```

```
1200  IF(NEWC.EQ.1)GO TO 1220
      CALL NEWCOM(PHRS,CMPNTS,ICOMP,LOCNO,CMPMOD,NOCOMP,
     1DINPUT,NDINPUT,DOUT,NDOUT,IDCOMP)
      DCOMNAM=PHRS
      NEWC=1
      GO TO 300
1220  WRITE(IWRITE,1221)DCOMNAM,PHRS
1221  FORMAT(/5X,28H *** WARNING *** COMPONENT  ,A10,*  DEFINITION WASN
     1T COMPLETED BEFORE STARTING THE DEFINITION OF COMPONENT  *,A10)
      ITASK=6
      GO TO 3000
C ====================  DIAGNOSTIC CONTROL    ITASK = 10
1400  CALL NXTPH(ICOM,INDEX,PHRS)
C ---       CHECK FOR NUMERIC IPUT, SKIP INPUT IF NOT NUMERIC
      CALL NUMERC(PHRS),RETURNS(300)
C ---       CONVERT TO INTEGER
      CALL BCDREL(PHRS,PHRS)
      IDIAG=PHRS
      GO TO 300
C ====================  INPUTS      ITASK = 2
C ---       TEST TO ASSURE THAT COMP. HAS BEEN IDENTIFIED.
2000  IF(LTASK.EQ.6)GO TO 300
C --->      ADD INPUTS TO COMPONENT
      CALL INCOM(ICOM,PHRS,INDEX,NDINPUT,DINPUT,NDOUT,DOUT,
     1 DCOMNAM,CMPMOD,NOCOMP,ICOMP)
      GO TO 320
C --->      STORE INPUT LIST FOR COMPONENT
3000  IF(IDCOMP.GE.1.AND.IDCOMP.LE.NOCOMP.AND.NDINPUT.GT.0)
     1 CALL WRITMS(7,DINPUT,NDINPUT,IDCOMP)
      NEWC=0
      GO TO 420
C ====================  END OF MODEL   COMPILE   ITASK = 4,5
C --->      FORM MODEL SUBROUTINES
4000  CALL ENDMOD(CMPMOD,NOCOMP,DOUT)
      GO TO(300,300,300,300,6200,300,6000,100,5900,1400,
     1 300,300,300,300,300),ITASK
C --->      WRITE FORTRAN ONTO SOURCE FILE
5000  WRITE(9,101)ICOM
      GO TO 100
C ====================  ADD STATES     ITASK = 11
C --->      ADD STATES TO MODEL
5100  WRITE(8,101)PHRS
      NOX=NOX+1
      GO TO 300
C ====================  ADD VARIABLES   ITASK = 12
C --->      ADD VARIABLES TO MODEL
5200  WRITE(11,101)PHRS
      NOV=NOV+1
      GO TO 300
C ====================  ADD PARAMETERS   ITASK = 13
C --->      ADD PARAMETERS TO MODEL
5300  WRITE(10,101)PHRS
      NOP=NOP+1
      GO TO 300
```

```
C ==================  ADD TABLES    ITASK = 14
C --->       ADD TABLES TO MODEL
C --->       GET TABLE DIMENSION IN NEXT PHRASE
5400  CALL NXTPH(ICOM,INDEX,TABDIM)
C --->       TEST TO ASSURE THAT TABLE DIMENSION IS NUMERIC
      CALL NUMERC(TABDIM),RETURNS(5420)
C --->       CONVERT TABLE DIMENSION TO INTEGER
      CALL BCDREL(TABDIM,TABDIM)
      I=TABDIM
      CALL PUTCOD(5,PHRS,I)
      NOTAB=NOTAB+1
      TABNAM(NOTAB)=PHRS
      GO TO 300
5420  WRITE(IWRITE,5421)PHRS,TABDIM
5421  FORMAT(/5X,29H *** WARNING *** TABLE NAME  ,A7,
     1* MUST BE FOLLOWED BY A NUMERIC DIMENSION RATHER THAN  *,A7)
      PHRS=TABDIM
      GO TO 320
C --->       SET INDICATOR TO PUNCH SOURCE DECKS
C ==================  PUNCH    ITASK =9
5900  IPUNCH=1
C ==================  PRINT    ITASK = 7
C --->       DRAW SCHEMATIC DIAGRAM
6000  CALL SCHEMA(CMPMOD,NOCOMP,DINPUT,DOUT)
C --->       PRINT INPUT REQUIREMENTS LIST
      REWIND 12
      WRITE(IWRITE,6161)
6161  FORMAT(1H1)
6170  READ(12,101)SOURCE
      IF(EOF(12))6200,6180
6180  WRITE(IWRITE,6181)SOURCE
6181  FORMAT(1X,7A10,A2)
      GO TO 6170
C --->       PUNCH SOURCE FILE
6200  IF(IPUNCH.NE.1)GO TO 100
      REWIND 9
6220  READ(9,101)SOURCE
      IF(EOF(9))100,6250
6250  WRITE(3,101)SOURCE
      GO TO 6220
6260  STOP
C ==================  LIST STANDARD COMPONENTS   ITASK = 15
6300  CALL LISTSC(ICPMAX,CMPNTS,DINPUT,DOUT)
      GO TO 300
C ==================  O.C. COMMANDS  ITASK = 16,17,18,19,20,22
C --->       INTERPRETE OPTIMAL CONTROLLER INPUTS
7000  CALL OCINTR(ITASK,PHRS)
      GO TO 300
C ==================  O.C. ANALYSIS ONLY   ITASK = 21
C --->       SET ANALYSIS ONLY FLAG
7100  IOCAN=1
      GO TO 300
      END
```

```
CENDCOM
      SUBROUTINE ENDCOM(AINPUT,COMNAM,SOURCE,ISOUR,NOCOMP,NSEQ)
C  VERSION 2.                          REVISED: DEC 15 1975
C  PURPOSE:  TO COMPLETE A COMPONENT DESCRIPTION IN THE ECS MODEL.
C  CALL SEQUENCE  AINPUT - LIST OF INPUT QUANTITY NAMES
C                 COMNAM - SPECIFIC COMPONENT NAME
C                 SOURCE - BUFFER ARRAY OF SOURCE CODE
C                 ISOUR  - INDEX TO NEXT CHARACTER IN SOURCE BUFFER
C                 NOCOMP - MODEL COMPONENT NO.
C                 NSEQ   - MODEL COMPONENT SEQUENCE NO.
      DIMENSION AINPUT(1),SOURCE(8)
      COMMON/CIO/IREAD,IWRITE,IDIAG
      COMMON /CORDER/NOX,NOV,NOP
      DATA COMMA/10H,         /,RPAR/10H)         /,BLNK/10H          /
      CALL GETCOD(5,COMNAM,NINPUT)
C ---       TEST FOR COMPONENTS WITH NO INPUTS
      IF(NINPUT.LE.0)GO TO 110
      CALL READMS(7,AINPUT,NINPUT,NOCOMP)
C --->      SCAN INPUTS
      DO 200 I=1,NINPUT
C --->      TEST 4TH CHARACTER TO DETERMINE IF INPUT SOURCE HAS BEEN SAT
      CALL GETT(AINPUT(I),4,CHAR)
      IF(CHAR.NE.BLNK) GO TO 100
C --->      NOT STAISFIED - TYPE INPUT AS A PARAMETER
C --->      FORM UNIQUE NAME BY ADDING COMPONENT NAME
      CALL NAMGEN(AINPUT(I),COMNAM,AINPUT(I))
C --->      INCREASE PARAMETER COUNTER
      NOP=NOP+1
C --->      ADD NAME TO PARAMETER NAME LIST
      WRITE(10,11)AINPUT(I)
11    FORMAT(A10)
C --->      ADD INPUT TO COMPONENT CALL SEQUENCE
100   CALL LINE(0,SOURCE,ISOUR,COMMA,1,12)
      CALL LINE(0,SOURCE,ISOUR,AINPUT(I),7,12)
200   CONTINUE
C --->      COMPLETE CALL SEQUENCE WITH )
110   CALL LINE(0,SOURCE,ISOUR,RPAR,1,12)
      IF(IDIAG.GE.50)WRITE(IWRITE,101)SOURCE
101   FORMAT(* ENDCOM-SOURCE*/(1X,6A10))
C --->      WRITE LINE ON SOURCE FILE
      WRITE(12,201)SOURCE
201   FORMAT(8A10)
C --->      GENERATE STATEMENT NUMBER
      NO=NSEQ+9000
C --->      WRITE CONTINUE STATEMENT ON SOURCE FILE
      WRITE(12,205)NO
205   FORMAT(1X,I4,1X,*CONTINUE*)
      RETURN
      END
```

```
CENDMOD
      SUBROUTINE ENDMOD(CMPMOD,NOCOMP,OUTPUT)
C   VERSION 4.1                                REVISED  JUNE 28 1977
C   PURPOSE:  COMPLETE THE GENERATION OF ECS MODEL SUBROUTINES EQMO   DAT
C   CALL SEQUENCE:  CMPMOD - ARRAY CONTAINING NAMES OF MODEL COMPS.
C                   NOCOMP - COMPONENT COUNTER
C                   OUTPUT - INPUT-OUTPUT-TABLE NAME ARRAY WORK SPACE
C    DESIGNED BY: J.D.BURROUGHS                              DATE: JULY 1974
      COMMON/CORDER/NOX,NOV,NOP/CTITLE/TITLE(7)/CSEQ/NSEQ,SEQA(1)
     1 /CTAB/NOTAB,TABNAM(1)/COC/NOCIN,NOCOUT,NOC,NOCMOD,NOCCR,LOCCOC,
     2 IOCAN,IXOC
      COMMON/CIO/IREAD,IWRITE,IDIAG
      DIMENSION IVRSET(160),XSOUR(8),GT(2),CMPMOD(1),OUTPUT(1)
      DATA GT/2OH      GO TO(         /,ECS/10HECS       /
      DATA CYCLES/12HCYCLES      /,DLINES/12HDLINES      /,
     1 RESET/12HRESET           /
      REWIND 12
      REWIND 9
C ---       GET PERMANENT FILE NAME
      CALL READMS(78,PFNAME,1,6HPFNAME)
      IF(IDIAG.EQ.21)WRITE(IWRITE,21)(CMPMOD(I),I=1,NOCOMP)
21    FORMAT(*  CMPMOD */10(1X,A10))
C --->      COMPLETE OPTIMAL CONTROLLER SPECIFICATION
      CALL OCEND(NOCOMP,CMPMOD,OUTPUT)
      IF(NOCOMP.LE.0)GO TO 90
C --->      CHECK COMPONENT SEQUENCE FOR IMPLICIT EQUATIONS
      CALL COMORD(CMPMOD,NOCOMP,OUTPUT)
C --->      SCAN MODEL COMPONENTS IN SEQUENCE OF LOCATION STATEMENTS
      DO 80 I=1,NSEQ
C --->      GET COMPONENT NO. IN LOCATION SEQUENCE
      CALL GETCOD(I,SEQA,ICOMP)
C --->      TEST FOR DIRECT FORTRAN COMPONENTS
      IF(CMPMOD(ICOMP).EQ.4HFORT)GO TO 60
      IF(I.EQ.1)WRITE(12,31)
   31 FORMAT(6X,*IF(CPUS.EQ.CPUSEC)GO TO 9000*,/6X,*ITEST=0*,
     1/6X,*IF(RESET.GT.0.)ITEST=1*,/6X,*CPUS=CPUSEC*,/6X,*ICNT=0*,
     2/6X,*IMPL=0*,/1X,*9000 CONTINUE*)
C --->      TEST FOR O.C. IF YES CALL OCCALL
      IF(KOMSTR(CMPMOD(ICOMP),1,2,2HOC,1).EQ.0)GO TO 72
C --->      INITIATE COMPONENT SUBROUTINE CALL GENERATION
      CALL CALLCP(CMPMOD(ICOMP),ICOMP,XSOUR,ISOUR,IVRSET,OUTPUT)
C --->      COMPLETE COMPONENT SUBROUTINE CALL GENERATION
      CALL ENDCOM(OUTPUT,CMPMOD(ICOMP),XSOUR,ISOUR,ICOMP,I)
      GO TO 80
C --->      TRANSFER DIRECT FORTRAN FROM FILE 9 TO FILE 12
60    READ(9,61)XSOUR
61    FORMAT(8A10)
C --->      TEST FOR EOF
      IF(EOF(9))80,70
70    IF(KOMSTR(XSOUR,1,4,4HFORT,1).EQ.0)GO TO 74
      WRITE(12,61)XSOUR
      GO TO 60
72    CALL OCCALL(CMPMOD,NOCOMP,I,IVRSET,OUTPUT)
   74 IF(I.EQ.1)WRITE(12,31)
80    CONTINUE
```

```
90     REWIND 9
C---              ADD PARAMETERS CYCLES,DLINES,RESET
C
       WRITE(10,81)CYCLES,DLINES,RESET
   81  FORMAT(A10)
       NOP=NOP+3
C ========>              FORM SUBROUTINE EQMO
       NOXP=MAXO(NOX,1)
       WRITE(9,91)TITLE,PFNAME,NOXP,NOXP
91     FORMAT(6X,*SUBROUTINE EQMO(TIME,TSTEP,INDP)*/*C*/*C*,9X,7A10/*C*/
      1*C --->       THIS SUBROUTINE WAS PREPARED BY THE EASY PRECOMPILER*
      2/*C*,25X,*USING  *,A10,* COMPONENTS*
      2/6X,*COMMON/CXDOT/XDOT(*,I4,*)/CINT/INT(*,I4,*)*
      3/6X,*COMMON/CSIMUL/DUM(6),TINC,TMAX,DUM2(6)*
      3/6X,*COMMON/CIMPL/IMPL,ICNT,ITEST/COVRLY/ADUM(3),CPUSEC*,
      4/6X,*COMMON/COST/CCO(9)*)
       IF(NOX.LT.1) GO TO 105
C --->       FORM /CX/ COMMON
       WRITE(9,93)
93     FORMAT(*C --->       STATE VARIABLES*)
       CALL COMGEN(NOX,2HCX,8,9)
105    IF(NOV.LT.1) GO TO 120
C --->       FORM /CV/ COMMON
       WRITE(9,111)
111    FORMAT(*C --->       VARIABLES*)
       CALL COMGEN(NOV,2HCV,11,9)
120    IF(NOP.LT.1) GO TO 140
C --->       FORM /CP/ COMMON
       WRITE(9,121)
121    FORMAT(*C --->       PARAMETERS*)
       CALL COMGEN(NOP,2HCP,10,9)
C --->       GENERATE TABLE COMMON IN EQMO
140    CALL TABGEN
C --->       GENERATE O.C. COMMONS
       IF(IOCAN.GT.0)CALL OCCOM
       WRITE(9,151)
151    FORMAT(*C --->                  MODEL EQUATIONS*)
C --->       TRANSFER CALL SEQUENCE FILE ONTO PROGRAM FILE
       REWIND 12
350    READ(12,61)XSOUR
       IF(EOF(12))400,370
370    WRITE(9,61)XSOUR
       GO TO 350
C --->       WRITE RETURN AND ENTRY VARSET AT END OF SUBROUTINE
400    WRITE(9,401)
401    FORMAT(6X,*CALL IMPLIC(CYCLES,DLINES)*,/6X,
      1*IF(CYCLES.LT.1.)GO TO 9900*,/6X,*IF(IMPL.LT.4)GOTO 9000*,
      2/6X,*IMPL=1*,/1X,*9900 CONTINUE*,/6X,*RETURN*,/6X,
      3*ENTRY VARSET*)
C --->       IVR = 2 FOR VARIABLES.  IVR = 0 FOR STATES.
       IVR=2
C --->       TEST THAT THERE ARE VARIABLES IN MODEL
       IF(NOV.LE.0) GO TO 620
C ---        TEST FOR MORE THAN 244 VARIABLES
       IF(NOV.GT.244)WRITE(9,411)IVR
```

```
411   FORMAT(6X,*IF(INDP.GT.244)GO TO 1000*,I1)
C --->      LOAD XSOUR WITH GO TO(
420   XSOUR(1)=GT(1)
      XSOUR(2)=GT(2)
      DO 500 I=3,8
500   XSOUR(I)=10H
      IXSOUR=13
      NGT=0
C --->      SCAN COMPONENTS
      DO 600 I=1,NOCOMP
C --->      GENERATE STATEMENT NO. CORRESPONDING TO EACH COMPONENT
      ISN=9000+I
C --->      CONVERT ISN TO BCD FORMAT
      ENCODE(4,501,ISN)ISN
501   FORMAT(I4)
C --->      INDEX FOR THE NO. OF VARIABLES (STATES) BEFORE COMPONENT WAS
      CALL GETCOD(I,SEQA,ICOMP)
      J=4*ICOMP-IVR-1
      CALL GETCOD(J,IVRSET,NO)
C --->      INDEX FOR THE NO. OF VARIABLES (STATES) AFTER COMPONENT WAS
      J=4*ICOMP-IVR
      CALL GETCOD(J,IVRSET,N1)
C --->      TEST TO DETERMINE IF ANY VARIABLES (STATES) WERE FORMED
      IF(N1.LE.NO) GO TO 600
      NO=NO+1
C --->      SCAN THE NO. OF VARIABLES (STATES) FOR THIS COMPONENT
      DO 520 J=NO,N1
      NGT=NGT+1
C ---       TEST IF 2ND LEVEL OF GO TO IS REQUIRED
      IF(NGT.LE.244)GO TO 515
      CALL LINE(0,XSOUR,IXSOUR,6H),INDP,6,9)
      WRITE(9,61)XSOUR
      WRITE(9,511)IVR
511   FORMAT(*1000*,I1,* INDP=INDP-244*)
      XSOUR(1)=GT(1)
      XSOUR(2)=GT(2)
      DO 505 K=3,8
505   XSOUR(K)=10H
      IXSOUR=13
      NGT=0
515   IF(IXSOUR.NE.13) CALL LINE(0,XSOUR,IXSOUR,1H,,1,9)
C --->      PLACE STATEMENT NO. IN COMPUTER GO TO STATEMENT
      CALL LINE(0,XSOUR,IXSOUR,ISN,4,9)
520   CONTINUE
600   CONTINUE
C --->      COMPLETE GO TO(   STATEMENT
      CALL LINE(0,XSOUR,IXSOUR,6H),INDP,6,9)
      WRITE(9,61)XSOUR
      IF(IVR.LE.0) GO TO 700
620   IVR=0
      WRITE(9,601)
601   FORMAT(6X,*ENTRY RATSET*)
C --->      TEST THAT THERE ARE STATES IN THE MODEL
      IF(NOX.LE.0) GO TO 700
C ---       TEST IF 2ND LEVEL OF GO TO IS REQUIRED FOR RATES
```

```
      IF(NOX.GT.244)WRITE(9,411)IVR
      GO TO 420
C ========>       FORM SUBROUTINE DATAIN       ====================
C ---->       COMMON AND DIMENSION STATEMENTS
700   WRITE(9,701)TITLE
701   FORMAT(6X,*END*////6X,*SUBROUTINE DATAIN*/*C*/*C*,9X,7A10/*C*/
     1*C ---->     THIS SUBROUTINE WAS PREPARED BY THE EASY PRECOMPILER*/
     26X,*COMMON/CORDER/NOX,NOV,NOP*)
C ---->       TEST IF STATES ARE PRESENT IN MODEL
      IF(NOX.LT.1) GO TO 715
C ---->       FORM STATE RELATED COMMONS
      WRITE(9,711)(NOX,I=1,10)
711   FORMAT(*C ---->       STATE RELATED COMMONS*/
     16X,*COMMON/CX/X(*,I4,*)/CXDOT/XDOT(*,I4,*)/CXIC/XIC(*,I4,*)*/
     25X,*1 /CXIC1/XIC1(*,I4,*)/CXIC2/XIC2(*,I4,*)/CXIC3/XIC3(*,I4,*)*/
     35X,*2 /CINT/INT(*,I4,*)/CNAMEX/NAMEX(*,I4,*)/CNAMER/NAMER(*,I4,*)*
     4/5X,*3 /CNTRLS/AN,IPRNT,MODE,ERROR(*,I4,*)*)
C ---->       CALCULATE THE AMOUNT OF WORK SPACE REQ D.
  715 NO=NOX*(2*NOX+7)
      IF(NO.LT.1000)NO=1000
      WRITE(9,719)NO
719   FORMAT(6X,*COMMON/CWORK/CWORK(*,I5,*)*)
C ---->       TEST IF VARIABLES ARE PRESENT IN MODEL
740   IF(NOV.LT.1) GO TO 780
      WRITE(9,741)NOV,NOV,NOV
741   FORMAT(*C ---->       VARIABLE RELATED COMMONS*/
     16X,*COMMON /CV/V(*,I4,*)/CNAMEV/NAMEV(*,I4,*)/COLD/VOLD(*,I4,*)*)
C ---->       TEST IF PARAMETERS ARE PRESENT IN MODEL
780   IF(NOP.LT.1) GO TO 800
      WRITE(9,781)NOP,NOP
781   FORMAT(*C ---->       PARAMETER RELATED COMMONS*/
     16X,*COMMON /CP/P(*,I4,*)/CNAMEP/NAMEP(*,I4,*)*)
C ---->       LOAD NO. OF STATE,VARIABLE, AND PARAMETERS INTO COMMONS
800   WRITE(9,821)NOX,NOV,NOP
821   FORMAT(*C ---->       SET NO. OF STATES,VARIABLES, AND PARAMETERS*/
     16X,*NOX=*,I4/6X,*NOV=*,I4/6X,*NOP=*,I4)
      IF(NOX.LE.0) GO TO 850
C ---->       LOAD STATE ERROR AND PARAMETER DEFAULT VALUES INTO COMMONS
      WRITE(9,831)
831   FORMAT(*C ---->       LOAD STATE ERROR DEFAULT VALUES*/
     16X,*DO 100 I=1,NOX*/6X,*ERROR(I)=.1*)
      IF(PFNAME.EQ.ECS)WRITE(9,833)
833   FORMAT(6X,*CALL GETT(NAMEX(I),1,KAR)*/6X,*IF(KAR.EQ.1HT)ERROR(I)=1
     3.*/6X,*IF(KAR.EQ.1HP)ERROR(I)=.005*)
      WRITE(9,841)
841   FORMAT(*100   CONTINUE*)
850   IF(NOP.LE.0) GO TO 860
      WRITE(9,851)
851   FORMAT(*C ---->       LOAD PARAMETER DEFAULT VALUES*/
     16X,*DO 300 I=1,NOP*/*300    P(I)=.99999*/
     26X,*WRITE(6,301)*/*301    FORMAT(1H1)*)
860   REWIND 12
C ---->       START FORMATION OF INPUT REQUIREMENTS LIST
      WRITE(12,861)TITLE,NOCOMP,NOTAB,NOP,NOX,NOV
861   FORMAT(//10X,7A10//5X,*THIS MODEL CONTAINS *,I4,* COMPONENTS*/
```

```
     15X,*WITH: *,I4,* TABLES*,2X,I4,* PARAMETERS*,2X,I4,* STATES AND*
     22X,I4,* VARIABLES.*
     2//10X,*INPUT DATA REQUIREMENTS LIST*/)
      MAXT=0
      IF(NOTAB.LE.0)GO TO 864
      CALL TABCAL
C =============       COMPLETE DATAIN SUBROUTINE  ==  START BLOCK DATA MODEL
C ---         CALCULATE TOTAL STORAGE REQUIRED BY MODEL TABLES
      DO 862 I=1,NOTAB
      CALL GETCOD(5,TABNAM(I),N)
      MAXT=MAXT+IABS(N)
862   CONTINUE
C ---         TESTS TO PREVENT DIMENSIONS < 1.
864   NOVP=MAX0(NOV,1)
      NOPP=MAX0(NOP,1)
      MAXTP=MAX0(MAXT,1)
      NOTABP=MAX0(NOTAB,1)
      WRITE(9,865)NOXP,NOVP,NOPP,MAXTP,NOTABP,NOTABP,NOTABP
865   FORMAT(6X,*RETURN*/6X,*END*////
     16X,*BLOCK DATA MODEL*/*C --->       MODEL NAME COMMONS*/
     26X,*COMMON/CNAMEX/NAMEX(*,I4,*)/CNAMEV/NAMEV(*,I4,
     3*)/CNAMEP/NAMEP(*,I4,*)*/5X,*1/CTABLE/TABLES(*,I4,*)/CTABNA/TABNAM
     4(*,I3,*)*/
     55X,*2/CMAXDI/NOTAB,MAXDIM(*,I3,*)/CLOCTA/LOCTAB(*,I3,*)*)
C --->       CREATE EQUIVALENCE STATEMENTS IF NEEDED TO ALLOW DATA
C --->       STATEMENTS TO LOAD NAME LISTS EXCEEDING 130 NAMES
      CALL COMEQU(5HNAMEX,NOX)
      CALL COMEQU(5HNAMEV,NOV)
      CALL COMEQU(5HNAMEP,NOP)
C ========  ADD FLIGHT CONDITION PARAMETER VALUES (ECS APPLICATION)
      IF(PFNAME.EQ.ECS)WRITE(9,866)
866   FORMAT(6X,*COMMON/AMISS/PAMB,TAMB,PRAM,TRAM,ALT,AMN*)
C --->       TEST FOR O.C.  IF YES CALL OCBLKD
      IF(IOCAN.GT.0)CALL OCBLKD
C --->       GENERATE NAME DATA STATEMENTS
      WRITE(9,867)
867   FORMAT(*C --->                  MODEL DATA STATEMENTS*)
C --->       GENERATE STATE, VARIABLE, AND PARAMETER NAME DATA STATEMENTS
      CALL NAMARY(5HNAMEX,5,NOX,8)
      CALL NAMARY(5HNAMEV,5,NOV,11)
      CALL NAMARY(5HNAMEP,5,NOP,10)
C --->       CALCULATE NO. OF WORDS IN TABLES
C ---         GENERATE TABLE NAMES, MAX DIMENSIONS,   LOCATIONS
      CALL TABDAT
C =======  ADD FLIGHT CONDITION PARAMETER VALUES (ECS APPLICATION)
      IF(PFNAME.EQ.ECS)WRITE(9,868)
868   FORMAT(6X,*DATA PAMB,TAMB,PRAM,TRAM,ALT,AMN/14.69,519.,14.69,519.,
     1 0.,0./*)
C ==========  TABLE INITIATION FOR ALL  MODELS
      WRITE(9,869)MAXTP
869   FORMAT(6X,*DATA TABLES/*,I5,9H*1.99999//6X,*END*///)
      IF(NOP.LE.0) GO TO 960
C --->       ADD PARAMETERS AND STATES TO INPUT REQUIREMENTS LIST
      NUNIT=10
      N1=NOP
      WRITE(12,881)
```

```
881    FORMAT(///14X,*PARAMETERS REQUIRED*//
      113X,*COMPONENT*,5X,*PARAMETER*/
      215X,*NAME*,10X,*NAME*)
900    REWIND NUNIT
       COMPS=10H
       DO 940 I=1,N1
C --->       SCAN PARAMETER (STATE) LIST
       READ(NUNIT,901)ANAME
901    FORMAT(A7)
       CALL STRMOV(ANAME,4,4,COMP,1)
C --->       COMPARE CURRENT COMPONENT NAME WITH PREVIOUS NAME
       IF(COMPS.EQ.COMP) GO TO 920
       WRITE(12,911)
911    FORMAT(1H )
       COMPS=COMP
920    WRITE(12,921)COMP,ANAME
921    FORMAT(15X,A4,9X,A7)
940    CONTINUE
960    IF(NOX.LE.0) RETURN
       IF(NUNIT.EQ.8) RETURN
       NUNIT=8
       N1=NOX
       WRITE(12,961)
961    FORMAT(///18X,*STATES*/
      12X,*(INITIAL CONDITIONS AND ERROR CONTROLS REQUIRED)*//
      213X,*COMPONENT*,6X,*STATE*/15X,*NAME*,10X,*NAME*)
       GO TO 900
       END
```

```
CHLINE
      SUBROUTINE HLINE(PAGE,LINE,IN,IR)
C  PURPOSE:  ADD A HORIZONTAL CONNECTION LINE TO ECS SCHEMATIC
C  CALL SEQUENCE: PAGE - 13X56 ARRAY CONTAINING HOLLORITH
C                        REPRESENTATION OF A PAGE
C                LINE - LINE NO. FOR HORIZONTAL LINE
C                IN   - INPUT COMPONENT COL. LOCATION
C                IR   - RECEIVING COMPONENT COL. LOCATION
      DIMENSION PAGE(13,56)
C --->      IS INPUT COMP. ON LEFT OR RIGHT
      IF(IN.GE.IR)GO TO 100
      POINT=10H>
      I1=IN
      I2=IR
      GO TO 200
C --->      INPUT IS ON RIGHT
100   POINT=10H<
      I1=IR
      I2=IN
C --->      PLACE POINT ON RECEIVING END OF LINE
200   CALL PUTT(PAGE(1,LINE),IR,POINT)
C --->      ADD NO. OF SYMBOLS REQ D. TO SPAN COLUMNS
      DO 300 I=I1,I2
C --->      TEST TO PREVENT OVERWRITING POINTS
      IF(KOMSTR(PAGE(1,LINE),I,1,1H<,1).EQ.0)GO TO 300
      IF(KOMSTR(PAGE(1,LINE),I,1,1H>,1).EQ.0)GO TO 300
C --->      ADD HORIZONTAL LINE SYMBOL
      CALL PUTT(PAGE(1,LINE),I,1H=)
300   CONTINUE
      RETURN
      END
```

```
CIJBIT
      FUNCTION IJBIT(A,I,J,N)
C  VERSION 1.                          REVISED: AUG 7 1975
C  PURPOSE:  SET IJBIT EQUAL TO THE I J ELEMENT IN BINARY ARRAY A
C  CALL SEQUENCE:  A - N X N BINARY ARRAY
C                  I - ROW INDEX
C                  J - COLUMN INDEX
C                  N - COLUMN DIMENSION OF ARRAY
C  DESIGNED BY: J.D. BURROUGHS                    JULY 1975
      DIMENSION A(1)
      IBIT=I+(J-1)*N
      IWORD=(IBIT-1)/60+1
      LBIT=MOD(IBIT,60)
      I1=1
      ASHIFT=SHIFT(A(IWORD),LBIT)
      IJBIT=ASHIFT.AND.I1
      RETURN
      END
```

```
CIJBIT1
      SUBROUTINE IJBIT1(A,I,J,N)
C  VERSION 1.                    REVISED: AUG 7 1975
C  PURPOSE:  LOAD 1 IN I J LOACTION OF N BY N BINARY ARRAY A.
C  CALL SEQUENCE:  A - N X N BINARY ARRAY
C                  I - ROW INDEX
C                  J - COLUMN INDEX
C                  N - COLUMN DIMENSION OF ARRAY
C  DESIGNED BY: J.D. BURROUGHS                JULY 1975
      DIMENSION A(1)
      IBIT=I+(J-1)*N
      IWORD=(IBIT-1)/60+1
      LBIT=60-MOD(IBIT,60)
      I1=1
      I2=SHIFT(I1,LBIT)
      A(IWORD)=A(IWORD).OR.I2
      RETURN
      END
```

```
CINCOM
      SUBROUTINE INCOM(ICOM,PHRS,INDEX,NDINPUT,DINPUT,NDOUT,
     1DOUT,DCOMNAM,CMPMOD,NOCOMP,ICOMP)
C  VERSION 3.                              REVISED: JUNE 27 1977
C  PURPOSE:  PERFORM INPUT-OUTPUT CONNECTIONS BETWEEN STD. COMPS.
C  CALL SEQUENCE:  ICOM   - COMMAND STRING ARRAY
C                  PHRS   - CURRENT PHRASE (UPSTREAM COMP. NAME UPON ENT
C                  INDEX  - COMMAND STRING CHARACTER INDEX
C                  NDINPUT- NO. OF INPUTS FOR DOWNSTREAM COMP.
C                  DINPUT - LIST OF INPUT QUANTITY NAMES FOR DOWNSTREAM
C                           COMPONENT
C                  NDOUT  - NO. OF OUTPUTS FOR DOWNSTREAM COMP.
C                  DOUT   - LIST OF OUTPUT QUANTITY NAMES FOR DOWNSTREAM
C                  DCOMNAM- SPECITIC COMPONENT NAME OF DOWNSTREAM COMP.
C                  CMPMOD - LIST OF COMPONENTS IN CURRENT MODEL
C                  NOCOMP - NO. OF COMP. IN CURRENT MODEL
C                  ICOMP  - UPSTREAM COMP. TYPE
      COMMON /CIO/IREAD,IWRITE,IDIAG
      DIMENSION ICOM(8),DINPUT(1),DOUT(1),UINPUT(64),UOUT(64),CMPMOD(1)
C --->      NO. OF CONNECTIONS INDICATOR
      NOCON=0
C --->      SAVE UPSTREAM COMPONENT NAME
      UCOMNAM=PHRS
C --->      GET LIST OF UPSTREAM COMP. OUTPUTS
      CALL COMDAT(UCOMNAM,4HOUTP,NUOUT,UOUT)
C --->      SCAN COMP. IN CURRENT MODEL
      DO 100 I=1,NOCOMP
C --->      TEST TO SEE IF UPSTREAM COMP. HAS BEEN DEFINED
      IF(KOMSTR(CMPMOD(I),1,4,UCOMNAM,1).EQ.0)GO TO 120
100   CONTINUE
C --->      GET STD. INPUT LIST FOR UPSTREAM COMP.
      CALL COMDAT(UCOMNAM,4HINPT,NUINPUT,UINPUT)
C --->      STORE COMP. LOC.=-100, COMP TYPE, NO. INPUTS FOR UPSTREAM CO
      CALL PUTCOD(3,UCOMNAM,-100)
      CALL PUTCOD(5,UCOMNAM,NUINPUT)
C --->      INCREMENT MODEL COMP. COUNT
      NOCOMP=NOCOMP+1
C --->      ADD COMP. NAME TO CURRENT MODEL LIST
      CMPMOD(NOCOMP)=UCOMNAM
      IUCOMP=NOCOMP
      GO TO 140
C --->      GET INPUT LIST FOR EXISTING COMP.
120   IUCOMP=I
      CALL GETCOD(5,CMPMOD(I),NUINPUT)
C --->      TEST FOR COMPONENT DRIVING ITSELF
      IF(KOMSTR(UCOMNAM,1,4,DCOMNAM,1).EQ.0)GO TO 130
C --->      GET INPUT LIST FROM FILE 7
      UINPUT(1)=3HZZZ
      IF(NUINPUT.GT.0)CALL READMS(7,UINPUT,NUINPUT,IUCOMP)
      GO TO 140
C --->      LOAD UPSTREAM INPUTS FROM DOWNSTREAM INPUTS LIST
130   DO 135 I=1,NUINPUT
135   UINPUT(I)=DINPUT(I)
C --->      DEFAULT ON PORT DESIGNATION IS BLANK (UNIVERSAL PORT)
140   UPORT=1H
      DPORT=1H
```

```
      MODE=1
      CALL NXTPH(ICOM,INDEX,PHRS)
      IPHRS=1
      IF(KOMSTR(PHRS,1,1,1H ,1).EQ.0)GO TO 500
C --->      TEST FOR NUMERIC, I.E. PORT NUMBER
      CALL NUMERC(PHRS),RETURNS(180)
C --->      SAVE NUMERIC PORT NO.
      MODE=1
      UPORT=PHRS
      CALL NXTPH(ICOM,INDEX,PHRS)
      IF(KOMSTR(PHRS,1,1,1H ,1).EQ.0)GO TO 160
C --->      TEST FOR NUMERIC, I.E. PORT NUMBER
      CALL NUMERC(PHRS),RETURNS(160)
C --->     SAVE DOWNSTREAM PORT NO.
      DPORT=PHRS
      IPHRS=0
      GO TO 420
160   WRITE(IWRITE,161)PHRS,UCOMNAM
161   FORMAT(/5X,18H *** WARNING ***  ,A10,*IS NOT A VALID PORT DESIGNAT
     1ION FOR INPUT COMPONENT  *,A4,*.   ERRONEOUS CONNECTIONS MAY OCCUR
     2*)
      GO TO 420
C --->      SCAN UPSTREAM OUTPUTS
180   DO 200 I=1,NUOUT
      IF(KOMSTR(UOUT(I),1,3,PHRS,1).EQ.0)GO TO 220
200   CONTINUE
      GO TO 500
C --->      SAVE OUTPUT NAME
220   UOUTNAM=UOUT(I)
      MODE=0
      CALL NXTPH(ICOM,INDEX,PHRS)
      CALL NUMERC(PHRS),RETURNS(240)
C --->      SAVE UNSTREAM PORT NO.
      UPORT=PHRS
      CALL NXTPH(ICOM,INDEX,PHRS)
C --->      SCAN DOWNSTREAM INPUTS
240   DO 260 I=1,NDINPUT
      IF(KOMSTR(DINPUT(I),1,3,PHRS,1).EQ.0)GO TO 280
260   CONTINUE
      WRITE(IWRITE,261)PHRS,DCOMNAM
261   FORMAT(/5X,18H *** WARNING ***  ,A10,*IS NOT A VALID INPUT QUANTIT
     1Y OR PORT DESIGNATION FOR COMPONENT  *,A4)
      GO TO 560
280   DINNAM=DINPUT(I)
      CALL NXTPH(ICOM,INDEX,PHRS)
      CALL NUMERC(PHRS),RETURNS(300)
      DPORT=PHRS
      IPHRS=0
C --->      SEARCH FOR MATCH BETWEEN NAMES   PORT NO. GIVEN ABOVE
300   DO 380 I=1,NDINPUT
C --->      TEST FOR NAME MATCH
      IF(KOMSTR(DINPUT(I),1,3,DINNAM,1).NE.0)GO TO 380
C --->      BYPASS PORT TEST IF PORT NOT SPECIFIED
      IF(DPORT.EQ.1H )GO TO 320
C --->      DOWNSTREAM PORT TEST
      IF(KOMSTR(DINPUT(I),9,1,DPORT,1).NE.0)GO TO 380
```

```
C --->      SCAN UPSTREAM OUTPUTS
320    DO 360 J=1,NUOUT
C --->      TEST FOR NAME MATCH
       IF(KOMSTR(UOUT(J),1,3,UOUTNAM,1).NE.0)GO TO 360
C --->      TEST IF PORT IS SPECIFIED
       IF(UPORT.EQ.1H )GO TO 400
C --->      TEST FOR PORT MATCH
       IF(KOMSTR(UOUT(J),9,1,UPORT,1).EQ.0)GO TO 400
360    CONTINUE
380    CONTINUE
       GO TO 560
C --->      SATISFY SPECIFIC INPUT
C --->      GET UPSTREAM AND DOWNSTREAM PORT NOS.
400    CALL GETT(UOUT(J),9,UPORT)
       CALL GETT(DINPUT(I),9,DPORT)
       CALL NAMGEN(UOUT(J),UCOMNAM,DINPUT(I))
C --->      TAG INPUT AS FROM AN UPSTREAM SOURCE
       CALL STRMOV(1H ,1,1,DINPUT(I),8)
       NOCON=1
       IF(MODE.EQ.0)GO TO 440
C --->      SATISFY ALL OTHER INPUTS USING OUTPUTS OF SPECIFIED PORTS
420    CALL PORTCN(DINPUT,NDINPUT,UOUT,NUOUT,DPORT,UPORT,UCOMNAM,NOCON,
      1 1H )
C --->      SATISFY UPSTREAM INPUTS
       CALL PORTCN(UINPUT,NUINPUT,DOUT,NDOUT,UPORT,DPORT,DCOMNAM,NOCON,
      1 1HD)
       GO TO 560
440    UPORT=1H
       DPORT=1H
       IF(IPHRS.EQ.1)GO TO 180
       CALL NXTPH(ICOM,INDEX,PHRS)
       IPHRS=1
       GO TO 180
500    IF(MODE.EQ.0)GO TO 560
C --->      REGULAR CONNECTION ROUTINE
C --->      SCAN DOWNSTREAM INPUTS
       DO 540 I=1,NDINPUT
C --->      TEST IF INPUT IS SATISFIED
       IF(KOMSTR(DINPUT(I),4,1,1H ,1).NE.0)GO TO 540
C --->      SCAN UPSTREAM OUTPUTS
       DO 520 J=1,NUOUT
C --->      TEST FOR NAME MATCH
       IF(KOMSTR(DINPUT(I),1,3,UOUT(J),1).EQ.0)GO TO 400
520    CONTINUE
540    CONTINUE
560    IF(NOCON.LE.0)WRITE(IWRITE,571)UCOMNAM,DCOMNAM
571    FORMAT(/5X,21H *** WARNING *** NO  ,A4,*  OUTPUTS MATCH UNSATISFIE
      1D  *,A4,*  INPUTS*)
C --->      STORE UPSTREAM INPUT LIST
       IF(NUINPUT.GT.0)CALL WRITMS(7,UINPUT,NUINPUT,IUCOMP)
       IF(IDIAG.LE.70)GO TO 600
       WRITE(IWRITE,801)(UINPUT(I),I=1,NUINPUT)
801    FORMAT(* INCOM-UINPUTS*/(1X,6A10))
       WRITE(IWRITE,803)(UOUT(I),I=1,NUOUT)
```

```
803   FORMAT(* INCOM-UOUT*/(1X,6A10))
      WRITE(IWRITE,805)(DINPUT(I),I=1,NDINPUT)
805   FORMAT(* INCOM-DINPUT*/(1X,6A10))
      WRITE(IWRITE,807)(DOUT(I),I=1,NDOUT)
807   FORMAT(* INCOM-DOUT*/(1X,6A10))
C --->      TEST IF NEXT PHRASE HAS BEEN USED
600   IF(IPHRS.EQ.0)CALL NXTPH(ICOM,INDEX,PHRS)
      RETURN
      END
```

```
CLINE
      SUBROUTINE LINE(MODE,SOURCE,ISOUR,TEXT,N,NTAPE)
C  PURPOSE:  TO CONTROL THE FLOW OF SOURCE TEXT AND GENERATE
C            CONTINUES AS NEEDED TO STAY WITHIN COLUMNS 1 - 72
C  CALL SEQUENCE:  MODE   - MODE=0 -> NEW LINE IS STARTED BEGINING WITH
C                           MODE=1 -> TEXT IS SPLIT TO FIT EXACTLY 7-72
C                  ISOUR  - NEXT CHARACTER FOR WRITING
C                  TEXT   - NEW TEXT STRING
C                  N      - NO. OF CHARACTERS TO ADD
C                  NTAPE  - FILE TO WRITTEN TO
      DIMENSION SOURCE(8)
      DATA X/1OH     X    /,BLNK/1OH          /
C --->      TEST FOR END OF LINE
      IF(ISOUR+N.LE.73) GO TO 300
      IF(MODE.NE.0) GO TO 400
C --->      NEW LINE REQUIRED
C --->      WRITE CURRENT LINE
      WRITE(NTAPE,101)SOURCE
101   FORMAT(8A10)
C --->      GENERATE CONTINUE SYMBOL
      SOURCE(1)=X
      DO 200 I=2,8
200   SOURCE(I)=BLNK
      ISOUR=7
300   CALL STRMOV(TEXT,1,N,SOURCE,ISOUR)
      ISOUR=ISOUR+N
      RETURN
C --->      MODE=1 SPLIT TEXT BETWEEN CURRENT AND NEXT LINE
400   NO=73-ISOUR
C --->      COMPLETE CURRENT LINE
      CALL STRMOV(TEXT,1,NO,SOURCE,ISOUR)
      WRITE(NTAPE,101)SOURCE
      SOURCE(1)=1OH     X
      DO 420 I=2,8
420   SOURCE(I)=1OH
C --->      NO. CHARACTERS LEFT IN TEXT
      L=N-NO
C --->      NEXT CHARACTER IN TEXT TO MOVE
      NO=NO+1
      CALL STRMOV(TEXT,NO,L,SOURCE,7)
      ISOUR=L+7
      RETURN
      END
```

```
CLISTSC
      SUBROUTINE LISTSC(ICPMAX,CMPNTS,AINPUT,OUTPUT)
C  VERSION 2.                          REVISED: OCT 8 1976
C  PURPOSE:  PROVIDE A LIST OF STANDARD COMPONENTS AND THEIR
C            INPUTS, OUTPUTS, AND TABLES
C  CALL SEQUENCE:  ICPMAX - NO. OF STANDARD COMPONENTS
C                  CMPNTS - LIST OF STANDARD COMPONENT NAMES
C                  AINPUT - WORK SPACE FOR INPUT NAMES
C                  OUTPUT - WORK SPACE FOR OUTPUT NAMES
      COMMON/CIO/IREAD,IWRITE,IDIAG
      DIMENSION CMPNTS(1),AINPUT(1),OUTPUT(1),TABLE(10)
      CALL READMS(78,PFNAME,1,6HPFNAME)
      WRITE(IWRITE,101)PFNAME
101   FORMAT(1H1,14X,*LIST OF STANDARD  *,A10,* COMPONENTS*)
C --->      SCAN STD. COMPONENTS
      DO 560 I=1,ICPMAX
      WRITE(6,521)I,CMPNTS(I)
521   FORMAT(///15X,*COMPONENT NO.*,I3,*   NAME = *,A2//
     13X,*INPUTS*,8X,*OUTPUTS*,16X,*TABLES*/
     22(* NAME  PORT    *),*  NAME   INDP. VAR.  MAX. DATA*)
C --->      GET INPUT,OUTPUT,AND TABLE NAMES
      CALL COMDAT(CMPNTS(I),4HINPT,NI,AINPUT)
      CALL COMDAT(CMPNTS(I),4HOUTP,NO,OUTPUT)
      CALL COMDAT(CMPNTS(I),4HTABS,NT,TABLE)
      MAX=MAX0(NI,NO,NT)
C --->      SCAN LONGEST LIST OF NAMES
      DO 560 J=1,MAX
C --->      BLANK NAMES
      AIN=10H
      OUT=10H
      TAB=10H
      ID=10H
      IP=10H
      OP=10H
      IV=10H
      ST=10H
      IF(J.GT.NI)GO TO 530
      AIN=AINPUT(J)
      CALL GETT(AIN,9,IP)
530   IF(J.GT.NO)GO TO 535
      OUT=OUTPUT(J)
      CALL GETT(OUT,9,OP)
      CALL GETT(OUT,10,ST)
535   IF(J.GT.NT)GO TO 540
      TAB=TABLE(J)
C --->      GET TABLE DIMENSION
      CALL GETCOD(5,TAB,ID)
      IV=1H2
      IF(ID.GT.0)GO TO 540
      IV=1H1
      ID=IABS(ID)
540   WRITE(IWRITE,541)AIN,IP,OUT,OP,ST,TAB,IV,ID
541   FORMAT(2X,A6,A1,8X,A6,A1,1X,A1,7X,A6,5X,A1,9X,I3)
560   CONTINUE
      WRITE(IWRITE,563)
563   FORMAT(1H1)
      RETURN
      END
```

```
CNAMARY
      SUBROUTINE NAMARY(CNAME,NCHAR,N,NUNIT)
C  VERSION 1.2                    REVISED: AUG 22 1975
C  PURPOSE: FORM A DATA STATEMENT THAT CONTAINS A GIVEN LIST OF NAMES
C  CALL SEQUENCE:  CNAME  - NAME OF THE ARRAY TO BE INITIALIZED
C                  NCHAR  - NO. OF CHARACTERS IN ARRAY NAME
C                  N      - NO. OF NAMES TO BE PLACED IN DATA STATEMENT
C                  NUNIT  - UNIT CONTAINING LIST OF NAMES
C  DESIGNED BY: J.D. BURROUGHS                MAY 1974
      DIMENSION SOURCE(8)
C --->      TEST FOR EMPTY SET
      IF(N.LE.0) RETURN
      REWIND NUNIT
C ---       CALCULATE THE NO. OF DATA STATEMENT EXTENSIONS REQD.
      NEXT=(N-1)/130+1
C ---       SCAN DATA STATEMENT EXTENSIONS
      DO 400 J=1,NEXT
C --        EXTENSION COUNTER
      K=J-1
C ---       NO. OF CHARACTERS PER EXTENSION
      N10=10*(N-K*130)
C ---       LIMIT NO. OF CHARACTERS PER DATA STATEMENT TO 1300
      IF(N10.GT.1300)N10=1300
C ---       CALC. FIRST AND LAST WORD IN LIST OF DATA STATEMENT
      ISTART=K*130+1
      ISTOP=ISTART+N10/10-1
C --->      GENERATE DATA STATEMENT
      SOURCE(1)=10H      DATA
      ISOUR=12
      DO 100 I=2,8
100   SOURCE(I)=10H
C --->      LOAD ARRAY NAME
      CALL LINE(0,SOURCE,ISOUR,CNAME,NCHAR,9)
C ---       TEST IF DATA STATEMENT EXTENSION IS REQUIRED
      IF(K.LE.0)GO TO 110
C ---       ENCODE DATA EXTENSION NO.
      ENCODE(2,105,K)K
105   FORMAT(I2)
C ---       ADD EXTENSION NO. TO DUMMY ARRAY NAME
      CALL LINE(0,SOURCE,ISOUR,K,2,9)
110   CALL LINE(0,SOURCE,ISOUR,1H/,1,9)
      ENCODE(4,121,N10)N10
121   FORMAT(I4)
C --->      LOAD NO. OF CHARACTERS IN DATA STATEMENT
      CALL LINE(0,SOURCE,ISOUR,N10,4,9)
      CALL LINE(0,SOURCE,ISOUR,1HH,1,9)
C --->      SCAN NAMES
      DO 200 I=ISTART,ISTOP
      READ(NUNIT,125)ANAME
125   FORMAT(A7,3H   )
C --->      LOAD NAMES INTO DATA STATEMENT
      CALL LINE(1,SOURCE,ISOUR,ANAME,10,9)
200   CONTINUE
      CALL LINE(1,SOURCE,ISOUR,1H/,1,9)
      WRITE(9,201)SOURCE
201   FORMAT(8A10)
400   CONTINUE
      RETURN
      END
```

```
CNAMGEN
      SUBROUTINE NAMGEN(SOURNM,COMNAM,QUANAM)
C  PURPOSE:  GENERATE UNIQUE NAMES FOR ALL MODEL VARIABLES PARAMETERS
C  CALL SEQUENCE:  SOURNM - SOURCE NAME
C                  COMNAM - COMPONENT NAME
C                  QUANAM - QUANTITY NAME
C ---->      TRANSFER SOURCE NAME TO QUANTITY NAME
      QUANAM=SOURNM
C --->       ADD COMP. NAME TO COL. 4 TO 7
      CALL STRMOV(COMNAM,1,4,QUANAM,4)
C ---        TEST COL. 9 FOR PORT NUMBER
      IF(KOMSTR(QUANAM,9,1,1H ,1).EQ.0)RETURN
C --->       TEST IF COL. 2 OR COL. 3 IS TO BE USED FOR PORT NO.
      I=3
      IF(KOMSTR(QUANAM,2,1,1H ,1).EQ.0)I=2
C --->       PLACE PORT NO. IN COL. I
      CALL STRMOV(QUANAM,9,1,QUANAM,I)
      RETURN
      END
```

```
CNEWCOM
      SUBROUTINE NEWCOM(COMNAM,CMPNTS,ICOMP,ALOC,CMPMOD,NOCOMP,
     1AINPUT,NINPUT,AOUT,NOUT,IDCOMP)
C  VERSION 2.                          REVISED: JAN 12 1976
C  PURPOSE:  INTRODUCE NEW COMPONENT INTO ECS MODEL
C  CALL SEQUENCE:  COMNAM - COMPONENT NAME
C                  CMPNTS - LIST OF STD. COMP. NAMES
C                  ICOMP  - LOCATION OF STD. COMP. NAME IN LIST
C                  ALOC   - COMPONENT LOCATION NO.
C                  CMPMOD - LIST OF COMP. IN CURRENT MODEL
C                  NOCOMP - NO. OF COMP. IN CURRENT MODEL
C                  AINPUT - STD. NAMES OF INPUTS FOR COMP.
C                  NINPUT - NO. OF INPUTS TO COMP.
C                  AOUT   - STD. NAMES OF OUTPUTS FOR COMP.
C                  NOUT   - NO. OF OUTPUTS FOR COMP.
C                  IDCOMP - COMP. NO. IN CURRENT MODEL
C    DESIGNED BY: J.D.BURROUGHS                          DATE: JULY 1974
      COMMON /CIO/IREAD,IWRITE,IDIAG/CSEQ/NSEQ,SEQA(1)
      DIMENSION CMPNTS(1),CMPMOD(1),AINPUT(1),AOUT(1)
C --->      CONVERT LOCATION NO. FROM HOLLORITH TO INTEGER
      CALL BCDREL(ALOC,ALOC)
      LOCNO=ALOC
C --->      GET SYMBOL NO. FOR COMPONENT AND PUT IN LOCATION 4
      CALL GETCOD(5,CMPNTS(ICOMP),ISYMB)
      CALL PUTCOD(4,COMNAM,ISYMB)
C --->      TEST THAT 1 OR MORE COMP. EXIST IN MODEL
      IF(NOCOMP.LE.0)GO TO 200
C --->      SCAN EXISTING COMPS. IN MODEL
      DO 100 I=1,NOCOMP
C --->      TEST THAT NEW COMP. NAME IS UNIQUE
      IF(KOMSTR(CMPMOD(I),1,4,COMNAM,1).EQ.0)GO TO 300
100   CONTINUE
C --->      NEW NAME IS UNIQUE
C --->      GET STD. INPUT LIST FOR COMP.
200   CALL COMDAT(COMNAM,4HINPT,NINPUT,AINPUT)
C --->      ADD LOC. NO. AND NO. OF INPUTS TO COMP. NAME
      CALL PUTCOD(3,COMNAM,LOCNO)
      CALL PUTCOD(5,COMNAM,NINPUT)
C --->      ADVANCE COMP. COUNT
      NOCOMP=NOCOMP+1
C --->      ADD NEW NAME TO MODEL COMP. NAME LIST
      CMPMOD(NOCOMP)=COMNAM
C --->      ADD COMP. NO. TO COMPONENT SEQUENCE LIST
      NSEQ=NSEQ+1
      CALL PUTCOD(NSEQ,SEQA,NOCOMP)
      IDCOMP=NOCOMP
C --->      GET LIST OF STD. OUTPUTS
220   CALL COMDAT(CMPNTS(ICOMP),4HOUTP,NOUT,AOUT)
      RETURN
C --->      TEST LOCATION NO. FOR COMP. THAT HAVE RECEIVED INPUTS BUT HA
C           BEEN DEFINED.
300   CALL GETCOD(3,CMPMOD(I),LN)
      IF(LN.LE.0)GO TO 400
      WRITE(IWRITE,301)COMNAM
301   FORMAT(/5X,29H *** WARNING ***  COMPONENT  ,A4,*  HAS ALREADY BEEN
     1DEFINED*)
      GO TO 420
```

```
C --->      ADD LOCATION NO. TO COMP. NAME
400   CALL PUTCOD(3,CMPMOD(I),LOCNO)
C ---       ADD SYMBOL NUMBER TO COMPONENT NAME
      CALL PUTCOD(4,CMPMOD(I),ISYMB)
C --->      ADD COMP. NO. TO COMPONENT SEQUENCE LIST
      NSEQ=NSEQ+1
      CALL PUTCOD(NSEQ,SEQA,I)
420   COMNAM=CMPMOD(I)
C --->      GET NO. OF INPUTS
      CALL GETCOD(5,COMNAM,NINPUT)
C --->      GET INPUT LIST FROM FILE 7
      AINPUT(1)=3HZZZ
      IF(NINPUT.GT.0)CALL READMS(7,AINPUT,NINPUT,I)
      IDCOMP=I
      GO TO 220
      END
```

```
CORDER
      SUBROUTINE ORDER(NV,ICO,A,IW1,IW2,IERROR,IB,IE)
C  VERSION 1.                    REVISED: AUG 4 1975
C  PURPOSE: GENERATE A SEQUENCE VECTOR THAT REORDERS VARIABLES
C           SO THAT CONNECTION MATRIX IS LOWER TRIANGULAR.
C  CALL SEQUENCE:  NV     - SYSTEM ORDER
C                  ICO    - SEQUENCE VECTOR
C                  A      - SYSTEM CONNECTION MATRIX
C                  IW1    - NTH ORDER VECTOR - PROCESS CODE
C                  IW2    - NTH ORDER VECTOR - PROCESS SEQUENCE
C                  IERROR - ERROR FLAG: 0 = SYSTEM WAS REDUCED TO LOWER
C                                         TRIANGULAR FORM.
C                                       1 = SYSTEM CAN NOT BE REDUCED T
C                                         TRIANGULAR FORM
C                  IB     - FIRST WORD IN IW2 POINTING TO LOOP COMP.
C                  IE     - LAST WORD IN IW2 POINTING TO LOOP COMP.
C  DESIGNED BY: F FATH                              JULY 1975
      DIMENSION ICO(1),IW1(1),IW2(1),A(1)
      NCO=0
      IERROR=0
C     SET ELEMENT COUNT IN PROCESS SEQUENCE VECTOR TO ZERO
      NTW2=0
C     INITIALIZE PROCESS CODE FOR EACH ELEMENT TO -1 (NO PROCESS)
      DO 10 I=1,NV
10    IW1(I)=-1
C     FIND FIRST NON-PROCESSED ELEMENT
15    DO 20 I=1,NV
      IF(IW1(I).LT.0)GO TO 30
20    CONTINUE
C     IF ALL ELEMENTS PROCESSED, RETURN
      RETURN
C     PUT NON-PROCESSED ELEMENT INTO PROCESS SEQUENCE VECTOR AT BOTTOM
30    NTW2=NTW2+1
      IW2(NTW2)=I
C     SET PROCESS CODE TO 0 (PARTIAL PROCESS)
      IW1(I)=0
C     CHECK FOR DEPENDANCE ON OTHER ELEMENTS
      JS=0
40    JS=JS+1
C     IF ALL ELEMENT DEPENDANCIES CHECKED, PROCESS IS COMPLETE
      IF(JS.GT.NV)GO TO 70
      K=IJBIT(A,I,JS,NV)
C     IF NO DEPENDANCE (K=0) KEEP LOOKING
      IF(K.EQ.0)GO TO 40
C     IF DEPENDANT ON ELEMENT ALREADY PROCESSED (CODE=1) KEEP LOOKING
C     IF DEPENDANT ON ELEMENT NOT PROCESSED (CODE=-1) START PROCESSING
C     ON THAT ELEMENT.
C     IF DEPENDANT ON ELEMENT PARTIALLY PROCESSED (CODE=0) SEQUENCING
C     IS IMPOSSIBLE.  SET ERROR FLAG AND START ERROR REPORT.
      IF(IW1(JS))50,60,40
50    I=JS
      GO TO 30
60    IERROR=1
C     LOOK FOR JS IN IW2.  THIS IS BEGINING OF DEPENDANT LOOP
      DO 65 K=1,NTW2
      IF(IW2(K).EQ.JS)GO TO 66
```

```
65    CONTINUE
66    IB=K
C     SET END OF LOOP POINTER
      IE=NTW2
C     RETURN DUE TO ERROR
      RETURN
C     PROCESS FOR ELEMENT COMPLETE - UPDATE PROCESSED ELEMENT COUNT
70    NCO=NCO+1
C     SET SEQUENCE VECTOR POSITION TO INDICATE ELEMENT
      ICO(NCO)=I
C     SET PROCESS CODE FOR ELEMENT TO COMPLETE (CODE=1)
      IW1(I)=1
C     DECREMENT PROCESS SEQUENCE POINTER
      NTW2=NTW2-1
C     IF ALL PROCESSED - RETURN
      IF(NCO.EQ.NV)RETURN
C     IF NO ELEMENT LEFT IN PROCESS SEQUENCE VECTOR, GO LOOK FOR FIRST
C     NON-PROCESSED ELEMENT.
      IF(NTW2.LE.0)GO TO 15
C     CONTINUE PROCESSING BOTTOM ELEMENT IN PROCESS SEQUENCE VECTOR
C     WHERE IT WAS INTERRUPTED.
      JS=I
      I=IW2(NTW2)
      GO TO 40
      END
```

```
CPORTCN
      SUBROUTINE PORTCN(AINPUT,NINPUT,OUTPUT,NOUT,IPORT,OPORT,OUTNAM,
     1 NOCON,STREAM)
C  PURPOSE:  CONNECT ALL MATCHING PHYSICAL QUANTITIES AT SPECIFIED
C            PORTS ON TWO COMPONENTS.
C  CALL SEQUENCE:  AINPUT - INPUT QUANTITY NAME LIST
C                  NINPUT - NO. OF INPUT QUANTITIES
C                  OUTPUT - OUTPUT QUANTITY NAME LIST
C                  NOUT   - NO. OF OUTPUT QUANTITIES
C                  IPORT  - INPUT PORT NO.
C                  OPORT  - OUTPUT PORT NO.
C                  OUTNAM - OUTPUT COMP. NAME
C                  NOCON  - NO CONNECTION FLAG
C                  STREAM - SOURCE INDICATOR. BLANK = UPSTREAM SOURCE
C                           D = DOWNSTREAM SOURCE
      DIMENSION AINPUT(1),OUTPUT(1)
C --->      SCAN INPUT LIST
      DO 200 I=1,NINPUT
C --->      TEST IF INPUT IS SATISFIED
      IF(KOMSTR(AINPUT(I),4,1,1H ,1).NE.0)GO TO 200
C --->      BYPASS PORT TEST IF INPUT IS UNIVERSAL PORT
      IF(KOMSTR(AINPUT(I),9,1,1H ,1).EQ.0)GO TO 100
C --->      BYPASS TEST IF SPECIFIED PORT IS UNIVERSAL PORT
      IF(IPORT.EQ.1H )GO TO 100
C --->      COMPARE PORTS
      IF(KOMSTR(AINPUT(I),9,1,IPORT,1).NE.0)GO TO 200
C --->      SCAN OUTPUTS
100   DO 120 J=1,NOUT
C --->      TEST FOR PHYSICAL QUANTITY MATCH
      IF(KOMSTR(AINPUT(I),1,3,OUTPUT(J),1).NE.0)GO TO 120
C --->      BYPASS PORT TEST IF SPECIFIED PORT IS UNIVERSAL PORT
      IF(OPORT.EQ.1H )GO TO 140
C --->      BYPASS PORT TEST IF OUTPUT IS UNIVERSAL PORT
      IF(KOMSTR(OUTPUT(J),9,1,1H ,1).EQ.0)GO TO 140
C --->      TEST FOR PORT MATCH
      IF(KOMSTR(OUTPUT(J),9,1,OPORT,1).EQ.0)GO TO 140
120   CONTINUE
      GO TO 200
C --->      SATISFY INPUT
140   CALL NAMGEN(OUTPUT(J),OUTNAM,AINPUT(I))
C --->      PLACE SOURCE INDICATOR IN NAME
      CALL STRMOV(STREAM,1,1,AINPUT(I),8)
      NOCON=1
200   CONTINUE
      RETURN
      END
```

```
CSCHEMA
      SUBROUTINE SCHEMA(CMPMOD,NOCOMP,INPUTS,NAMES)
C  VERSION 2.                      REVISED: SEPT 10 1975
C  PURPOSE:  PRODUCE A SCHEMATIC DIAGRAM ON THE LINEPRINTER
C            OF THE ECS MODEL
C  CALL SEQUENCE:  CMPMOD - LIST OF COMPONENTS IN MODEL
C                  NOCOMP - NO. OF COMP. IN MODEL
C                  INPUTS - WORK ARRAY FOR INPUT NAMES
C                  NAMES  - WORK ARRAY FOR LABEL NAMES
C  DESIGNED BY: J.D. BURROUGHS                  JUNE 1974
      COMMON /CIO/IREAD,IWRITE,IDIAG/CTITLE/TITLE(7)
      DIMENSION PAGE(13,56),CMPMOD(1),INPUTS(1),NAMES(1)
      MAXPAG=0
      NPAGE=0
C --->      BLANK PAGE AND LOAD LOCATION NUMBERS
100   LOC=NPAGE
C --->      LOCATION NO. LINE COUNTER
      LOCL=4
C --->      SCAN ALL LINES ON PAGE
      DO 160 I=1,56
C --->      BLANK ENTIRE LINE
      DO 120 J=1,13
120   PAGE(J,I)=1OH
C --->      TEST IF LINE CONTAINS LOCATION NUMBERS.
      IF(I.LT.LOCL)GO TO 160
C --->      INCREMENT LOCATION NO. LINE COUNTER
      LOCL=LOCL+7
      LOCCOL=-8
C --->      SCAN COLS. AND LOAD LOCATION NOS.
      DO 140 J=1,10
C --->      INCREMENT LOCATION NO.
      LOC=LOC+1
      LOCCOL=LOCCOL+13
      ENCODE(4,139,LOCNO)LOC
139   FORMAT(I4)
      CALL STRMOV(LOCNO,1,4,PAGE(1,I),LOCCOL)
140   CONTINUE
160   CONTINUE
C --->      PLACE COMPONENT SYMBOLS ON PAGE
C --->      TEST THAT MORE THAN 0 COMP. EXIST IN MODEL
      IF(NOCOMP.LE.0)GO TO 602
C ---       SCAN COMPS. IN MODEL
      DO 300 I=1,NOCOMP
      COMNAM=CMPMOD(I)
C ---       SKIP FORTRAN COMPONENTS
      IF(KOMSTR(COMNAM,1,4,4HFORT,1).EQ.0)GO TO 300
C ---       GET LOCATION NO. FROM COMP. NAME
      CALL GETCOD(3,COMNAM,LOC)
C --->      DETERMINE PAGE NO.
      LOCPAG=(LOC/100)*100
C --->      DETERMINE MAX. NO. OF PAGES REQ D.
      MAXPAG=MAXO(MAXPAG,LOCPAG)
C --->      DETERMINE MAX. NO. OF PAGES REQ D.
      MAXPAG=MAXO(MAXPAG,LOCPAG)
C --->      TEST IF COMPONENT IS ON CURRENT PAGE
      IF(LOCPAG.NE.NPAGE)GO TO 300
```

```
C --->      CONVERT GENERAL PAGE LOCATION TO LOCAL PAGE LOCATION
      LOCPAG=LOC-LOCPAG
C --->      TEST TO ASSURE LOC NO. IS ON PAGE
      IF(LOCPAG.LT.1.OR.LOCPAG.GT.80)GO TO 260
C --->      ADD SYMBOL TO CURRENT PAGE FOR COMPONENT
      CALL GETCOD(4,COMNAM,ISYMB)
      IF(IDIAG.EQ.22)WRITE(IWRITE,251)COMNAM,COMNAM,ISYMB
251   FORMAT(* SCHEMA *,A10,1X,O20,I10)
      CALL SYMBOL(PAGE,COMNAM,ISYMB,LOCPAG)
C --->      FORM TABLE OF COMPONENT NAMES (ON ONLY FIRST PASS)
      GO TO 300
260   WRITE(IWRITE,261)LOC,COMNAM
261   FORMAT(/5X,31H *** WARNING ***  LOCATION NO. ,I4,
     1 * FOR COMPONENT  *,A4,*  HAS LAST TWO DIGITS OUTSIDE THE ALLOWABL
     2E RANGE OF 1 TO 80.*/18X,
     3*NO SYMBOL WILL BE PLACED IN SCHEMATIC FOR THIS COMPONENT.*)
      LOC=-100
300   CONTINUE
C --->      ADD CONNECTING LINES AND NAMES TO SCHEMATIC
C --->      SCAN MODEL COMPONENTS
400   DO 500 I=1,NOCOMP
C --->      BYPASS DIRECT FORTRAN INPUT COMPONENTS
      IF(KOMSTR(CMPMOD(I),1,4,4HFORT,1).EQ.0)GO TO 500
C --->      GET LOCATION NO.
      CALL GETCOD(3,CMPMOD(I),LOC)
C --->      DETERMINE PAGE NO.
      LOCPAG=(LOC/100)*100
C --->      CONVERT LOC TO LOCAL PAGE LOCATION
      LOC=LOC-LOCPAG
C --->      TEST TO ASSURE LOC NO. IS ON PAGE
      IF(LOC.LT.1.OR.LOC.GT.80)LOCPAG=-1
C --->      SKIP INPUTS TO QUANTITIES ON OTHER PAGES
      IF(LOCPAG.NE.NPAGE)GO TO 500
C --->      GET NO. OF INPUTS TO COMP.
      CALL GETCOD(5,CMPMOD(I),NINPUT)
C --->      BYPASS COMP. WITH NO INPUTS
      IF(NINPUT.LE.0)GO TO 500
C --->      GET INPUTS LIST
      CALL READMS(7,INPUTS,NINPUT,I)
C --->      INITIALIZE NO. INPUTS COUNTER   CURRENT INPUT COMP. NAME
420   NOIN=0
      MORE=0
      INCOM=7H*******
      IF(IDIAG.EQ.30)WRITE(IWRITE,423)CMPMOD(I),(INPUTS(J),J=1,NINPUT)
423   FORMAT(* SCHEMA-INPUTS  *,A10/10(1X,A10))
C --->      SCAN INPUTS
      DO 480 J=1,NINPUT
C --->      TEST IF INPUT IS FROM CURRENT COMP. I.E. PARAMETER
      IF(KOMSTR(INPUTS(J),4,1,1H ,1).EQ.0)GO TO 480
C --->      IS THIS A NEW INPUT SOURCE
      IF(KOMSTR(INCOM,4,4,INPUTS(J),4).EQ.0)GO TO 440
C ---       BYPASS NAME LOAD IF 2ND COMPONENT APPEARS
      IF(MORE.NE.0)GO TO 460
C --->      SAVE NEW SOURCE NAME
      INCOM=INPUTS(J)
      MORE=1
```

```
C --->      ADVANCE INPUT COUNT
440   NOIN=NOIN+1
      NAMES(NOIN)=INPUTS(J)
      INPUTS(J)=10H
      GO TO 480
460   MORE=2
480   CONTINUE
C --->      IS THERE A CURRENT INPUT COMPONENT
      IF(NOIN.LE.0)GO TO 500
      CALL CONNCT(PAGE,NPAGE,LOC,NAMES,NOIN,CMPMOD,NOCOMP)
C ---       DO MORE COMPONENTS PROVIDE INPUTS
      IF(MORE.EQ.2)GO TO 420
500   CONTINUE
C --->      PRINT PAGE
602   NAME=NPAGE/100
      WRITE(IWRITE,605)TITLE,NAME,PAGE
605   FORMAT(1H1,29X,7A10,24X,*PAGE*,I3/(2X,13A10))
C --->      TEST FOR LAST PAGE
      IF(NPAGE.GE.MAXPAG)RETURN
      NPAGE=NPAGE+100
      GO TO 100
      END
```

```
CSYMBOL
      SUBROUTINE SYMBOL(PAGE,COMNAM,ISYMB,LOC)
C  VERSION 1.2                        REVISED: OCT 17 1975
C  PURPOSE:  ADD COMPONENT SYMBOLS AND NAMES TO ECS MODEL SCHEMATIC
C  CALL SEQUENCE:  PAGE   - 13X56 ARRAY CONTAINING HOLLERITH
C                           REPRESENTATION OF A PAGE
C                  COMNAM - NAME OF COMPONENT TO BE ADDED TO PAGE
C                  ISYMB  - SYMBOL TYPE NO.
C                  LOC    - LOCATION OF SYMBOL ON PAGE
C  DESIGNED BY: J.D.BURROUGHS                    JUNE 1974
      COMMON/CIO/IREAD,IWRITE,IDIAG
      DIMENSION PAGE(13,56)
C --->      LOCATION LINE NO.
      LOCLIN=7*((LOC-1)/10)+3
C --->      LOCATION COLUMN  NO.
      LOCCOL=(MOD(LOC-1,10)+1)*13-10
C --->      ADD COMPONENT NAME TO PAGE
      CALL STRMOV(COMNAM,1,4,PAGE(1,LOCLIN),LOCCOL+3)
      IF(IDIAG.EQ.22)WRITE(IWRITE,22)COMNAM,ISYMB,LOC
22    FORMAT(* SYMBOL *,A10,2I10)
C --->      TEST FOR SYMBOL TYPE
C
C           SYMBOL NUMBERS LESS THAN 64 SHOULD NOT BE USED DUE TO
C           CSORT REPLACING 00B WITH 55B WHEN CALLED BY FILOAD.
C
      IF(ISYMB.EQ.100)GO TO 200
      IF(ISYMB.EQ.200)GO TO 400
      IF(ISYMB.EQ.300) GO TO 300
      IF(ISYMB.EQ.400)GO TO 500
C --->      DEFAULT SYMBOL - SQUARE
      LOCLIN=LOCLIN-2
C --->      TOP AND BOTTOM LINES
      CALL STRMOV(10H**********,1,10,PAGE(1,LOCLIN),LOCCOL)
      CALL STRMOV(10H**********,1,10,PAGE(1,LOCLIN+5),LOCCOL)
C --->      SIDES
      DO 100 I=1,4
      CALL PUTT(PAGE(1,LOCLIN+I),LOCCOL,1H*)
      CALL PUTT(PAGE(1,LOCLIN+I),LOCCOL+9,1H*)
100   CONTINUE
      RETURN
C --->      COMPRESSOR SYMBOL
200   L=LOCCOL
      K=2
      ICOL=L+1
205   LOCLIN=LOCLIN-5
      DO 220 I=1,10
      LOCLIN=LOCLIN+1
C --->      TEST TO PREVENT TOP OF SYMBOL FROM GOING OFF TOP OF PAGE
      IF(LOCLIN.LT.1)GO TO 208
C --->      TEST TO PREVENT BOTTOM OF SYMBOL FROM GOING OFF PAGE
      IF(LOCLIN.GT.56)RETURN
C --->      STRAIGHT EDGE OF SYMBOL
      CALL STRMOV(1H*,1,1,PAGE(1,LOCLIN),L)
C --->      SLOPING EDGE OF SYMBOL
      CALL STRMOV(1H*,1,1,PAGE(1,LOCLIN),ICOL)
```

```
C ==---->      TEST TO REVERSE SLOPE OF RIGHT EDGE
208    IF(I.EQ.5)GO TO 215
       ICOL=ICOL+K
       GO TO 220
215    K=-K
220    CONTINUE
       RETURN
C --->       TURBINE SYMBOL
300    L=LOCCOL+9
       K=-2
       ICOL=L-1
       GO TO 205
C --->       CIRCLE SYMBOL
400    LOCLIN=LOCLIN-2
       CALL STRMOV(10H  ******  ,1,10,PAGE(1,LOCLIN),LOCCOL)
       CALL STRMOV(10H  ******  ,1,10,PAGE(1,LOCLIN+5),LOCCOL)
       K=1
       L=LOCCOL+1
       ICOL=L+7
C --->       ADD SIDES TO SYMBOL
       DO 420 I=1,4
       LOCLIN=LOCLIN+1
C --->       LEFT EDGE OF SYMBOL
       CALL STRMOV(1H*,1,1,PAGE(1,LOCLIN),L)
C --->       RIGHT EDGE OF SYMBOL
       CALL STRMOV(1H*,1,1,PAGE(1,LOCLIN),ICOL)
C --->       REVERSE SLOPE OF EDGES
       IF(I.EQ.2)GO TO 415
       L=L-K
       ICOL=ICOL+K
       GO TO 420
415    K=-K
420    CONTINUE
       RETURN
C ---        OPTIMAL CONTROLLER SYMBOL
500    LOCLIN=LOCLIN-2
C --->       TOP AND BOTTOM LINES
       CALL STRMOV(10H 00000000 ,1,10,PAGE(1,LOCLIN),LOCCOL)
       CALL STRMOV(10H 00000000 ,1,10,PAGE(1,LOCLIN+5),LOCCOL)
C --->       SIDES
       DO 520 I=1,4
       CALL PUTT(PAGE(1,LOCLIN+I),LOCCOL,1HO)
       CALL PUTT(PAGE(1,LOCLIN+I),LOCCOL+9,1HO)
520    CONTINUE
       RETURN
       END
```

```
CTABCAL
      SUBROUTINE TABCAL
C  PURPOSE: GENERATE TABLE INPUT REQUIREMENTS LIST ON FILE 12
      COMMON/CTAB/NOTAB,TABNAM(1)
      WRITE(12,11)
11    FORMAT(16X,*TABLES REQUIRED*//
     12X,*COMPONENT   TABLE     NO. INDEP.    MAX. DATA*/
     24X,*NAME*,7X,*NAME*,5X,*VARIABLES     ALLOWED*)
      COMPS=10H
      COMP=COMPS
C --->      SCAN TABLES.
      DO 100 I=1,NOTAB
C --->      GET TABLE NAME
      CALL STRMOV(TABNAM(I),1,7,ANAME,1)
C --->      GET MAXIMUM DIMENSION FOR TABLE
      CALL GETCOD(5,TABNAM(I),N)
      N1=IABS(N)
C --->      GET SPECIFIC COMPONENT NAME
      CALL STRMOV(ANAME,4,4,COMP,1)
      IF(COMP.EQ.COMPS) GO TO 60
      WRITE(12,51)
51    FORMAT(1H )
      COMPS=COMP
60    N1=N1-3
C --->      TEST FOR SINGLE OR DOUBLE INDEP. VARIABLE TABLE
      IF(N.GT.0) GO TO 70
      N=1
      GO TO 80
70    N=2
80    WRITE(12,81)COMP,ANAME,N,N1
81    FORMAT(4X,A4,5X,A7,6X,I1,10X,I4)
100   CONTINUE
      RETURN
      END
```

```
CTABDAT
      SUBROUTINE TABDAT
C  VERSION 3.                          REVISED MAY 4 1976
C  PURPOSE:  GENERATE DATA STATEMENTS FOR MODEL TABLE DATA INPUT CONTROL
C   DESIGNED BY:  J.D.BURROUGHS                          DATE: MARCH 1975
      COMMON/CTAB/NOTAB,TABNAM(1)
      DIMENSION SOURCE(8)
C ==========  SET NUMBER OF TABLES IN MODEL
      WRITE(9,91)NOTAB
91    FORMAT(6X,*DATA NOTAB/*,I3,*/*)
      IF(NOTAB.LE.0)RETURN
C --------->      LOAD TABLE NAME DATA
      SOURCE(1)=10H      DATA
      SOURCE(2)=10H TABNAM/
      ISOUR=19
      DO 100 I=3,8
100   SOURCE(I)=10H
C --->      CALC. NO. OF CHARACTERS IN TABLE NAME LIST
      N10=10*NOTAB
      ENCODE(4,101,N10)N10
101   FORMAT(I3,1HH)
C --->      ADD NO. OF CHARACTERS TO DATA STATEMENT LINE
      CALL LINE(0,SOURCE,ISOUR,N10,4,9)
      ANAME=10H
C --->      SCAN TABLES
      DO 200 I=1,NOTAB
      CALL STRMOV(TABNAM(I),1,7,ANAME,1)
C --->      ADD TABLE NAME TO LINE
      CALL LINE(1,SOURCE,ISOUR,ANAME,10,9)
200   CONTINUE
      CALL LINE(1,SOURCE,ISOUR,1H/,1,9)
      WRITE(9,201)SOURCE
201   FORMAT(8A10)
C --------->         LOAD TABLE DIMENSION DATA
      SOURCE(1)=10H      DATA
      SOURCE(2)=10H MAXDIM/
      ISOUR=19
      DO 220 I=3,8
220   SOURCE(I)=10H
C --->      SCAN TABLES
      DO 240 I=1,NOTAB
C --->      GET MAX. TABLE DIMENSION
      CALL GETCOD(5,TABNAM(I),N)
      N=IABS(N)
C --->      CONVERT TO DISPLAY CODE
      ENCODE(5,231,N)N
231   FORMAT(I4,1H,)
      IF(I.GE.NOTAB)CALL STRMOV(1H/,1,1,N,5)
C --->      ADD MAX. DIMENSION TO LINE
      CALL LINE(0,SOURCE,ISOUR,N,5,9)
240   CONTINUE
      WRITE(9,201)SOURCE
```

```
C ---------->         LOAD TABLE LOCATION DATA
      SOURCE(1)=10H       DATA
      SOURCE(2)=10H LOCTAB/
      ISOUR=19
      DO 300 I=3,8
300   SOURCE(I)=10H
      LOC=1
C ---->       SCAN TABLES
      DO 320 I=1,NOTAB
C ---->       CONVERT TO DISPLAY CODE
      ENCODE(5,231,N)LOC
      IF(I.GE.NOTAB)CALL STRMOV(1H/,1,1,N,5)
C ---->       ADD TABLE LOCATION NO. TO LINE
      CALL LINE(0,SOURCE,ISOUR,N,5,9)
C ---->       GET MAX. DIMENSION OF TABLE
      CALL GETCOD(5,TABNAM(I),N)
C ---->       CALC. THE NEXT TABLE STARTING LOCATION
      LOC=LOC+IABS(N)
320   CONTINUE
      WRITE(9,201)SOURCE
      RETURN
      END
```

CTABGEN

```
      SUBROUTINE TABGEN
C  PURPOSE:  GENERATE THE TABLE COMMON FOR ECS MODEL
C  CALL SEQUENCE:  NTAB - TOTAL NO. OF TABLES REQ D BY MODEL
C  METHOD:  THE NAMES OF THE TABLES AND THEIR DIMENSIONS ARE STORED
C           IN TABNAM.  THE NAME IS STORED IN THE FIRST 7 CHARACTERS
C           OF EACH WORD AND THE DIMENSION IS STORED IN THE LAST 2
C           CHARACTERS VIA THE ROUTINE PUTCOD.
      COMMON/CTAB/NOTAB,TABNAM(1)
      DIMENSION SOURCE(8),SOTAB(2)
      IF(NOTAB.LE.0)RETURN
      WRITE(9,10)
10    FORMAT(*C --->       TABLES*)
      SOURCE(1)=10H      COMM
      SOURCE(2)=10HON/CTABLE/
      DO 100 I=3,8
100   SOURCE(I)=10H
      ISOUR=22
C ---       SCAN ALL TABLES IN THE MODEL
      DO 200 I=1,NOTAB
C --->      GET TABLE DIMENSION
      CALL GETCOD(5,TABNAM(I),N)
      N=IABS(N)
C ---       GET TABLE NAME
      CALL STRMOV(TABNAM(I),1,7,SOTAB,1)
C --->      CONVERT DIMENSION TO BCD
      ENCODE(6,105,N)N
105   FORMAT(1H(,I3,2H),)
C ---       REMOVE COMMA IF LAST TABLE
      IF(I.GE.NOTAB)CALL STRMOV(1H ,1,1,N,6)
      CALL STRMOV(N,1,6,SOTAB,8)
C ---       ADD TABLE NAME TO SOURCE LINE
      CALL LINE(0,SOURCE,ISOUR,SOTAB,13,9)
200   CONTINUE
      WRITE(9,201)SOURCE
201   FORMAT(8A10)
      RETURN
      END
```

```
CVLINE
      SUBROUTINE VLINE(PAGE,ICOL,IN,IR)
C  PURPOSE:  PAGE - 13X56 ARRAY CONTAINING HOLLORITH
C                   REPRESNETATION OF A PAGE
C              ICOL - COLUMN NO. OF LINE
C              IN   - LINE NO. OF INPUT COMPONENT
C              IR   - LINE NO. OF RECEIVING COMPONENT
      DIMENSION PAGE(13,56)
C --->      IS INPUT ABOVE OR BELOW
      IF(IN.GE.IR)GO TO 100
C --->      INPUT IS ABOVE
      POINT=1OHV
      I1=IN
      I2=IR
      GO TO 200
C --->      INPUT IS BELOW
100   POINT=1OHA
      I1=IR
      I2=IN
C --->      PLACE POINT ON RECEIVING END OF LINE
200   CALL PUTT(PAGE(1,IR),ICOL,POINT)
C --->      ADD NO. OF SYMBOLS REQ D. TO SPAN LINES
      DO 300 I=I1,I2
C --->      TEST TO PREVENT OVERWRITING POINTS
      IF(KOMSTR(PAGE(1,I),ICOL,1,1HA,1).EQ.0)GO TO 300
      IF(KOMSTR(PAGE(1,I),ICOL,1,1HV,1).EQ.0)GO TO 300
      CALL PUTT(PAGE(1,I),ICOL,1HI)
300   CONTINUE
      RETURN
      END
```

3.0 ANALYSIS PROGRAM DESCRIPTION

3.1 INTRODUCTION

The Analysis program accepts program commands which describe analyses to be performed on the given system model. Each analysis is then performed on the nonlinear system model that was created by the Model Generation program. Each analysis resides in an overlay which is brought into core to perform the requested analysis. The system model is placed in the root of the overlay structure since it is accessed by all of the analyses. The core requirements of the program have thereby been held constant as numerous analyses have been added to the program's capabilities. However, program core requirements do vary as a function of model size, growing as the square of the number of states in the model.

3.2 PROGRAM STRUCTURE

Figure 3.2-1 contains a macro flow diagram of the SIMWEST Analysis program. This flow diagram shows the principle tasks of the program. For each task, a statement number of the main, (NONSIM), program is given along with the name of the principle program that accomplishes that task.

The sequence of performing the various tasks depends on the analysis and data requests. As each analysis is performed, its outputs are generated on the lineprinter.

3.2.1 Overlay Structure

Figure 3.2-2 contains a diagram of the overlay structure of the Analysis program. The main program, (NONSIM), the system model, (EQMO, DATAIN, MODEL and standard component subroutines), and other frequently used routines reside in the main overlay, (0,0). Table 3.2-1 provides a brief description of each overlay.

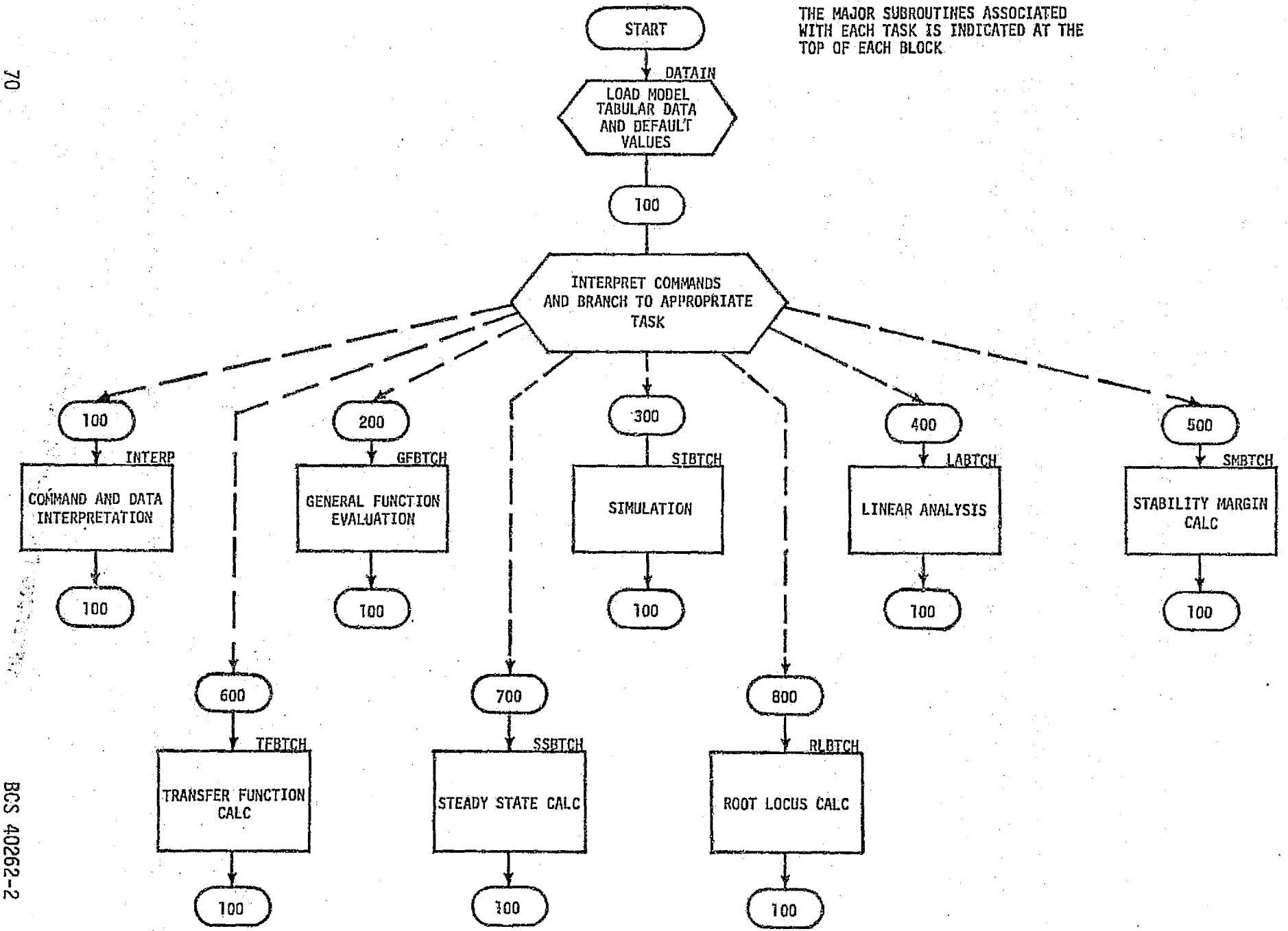

FIGURE 3.2-1 SIMWEST ANALYSIS PROGRAM - MACRO FLOW DIAGRAM

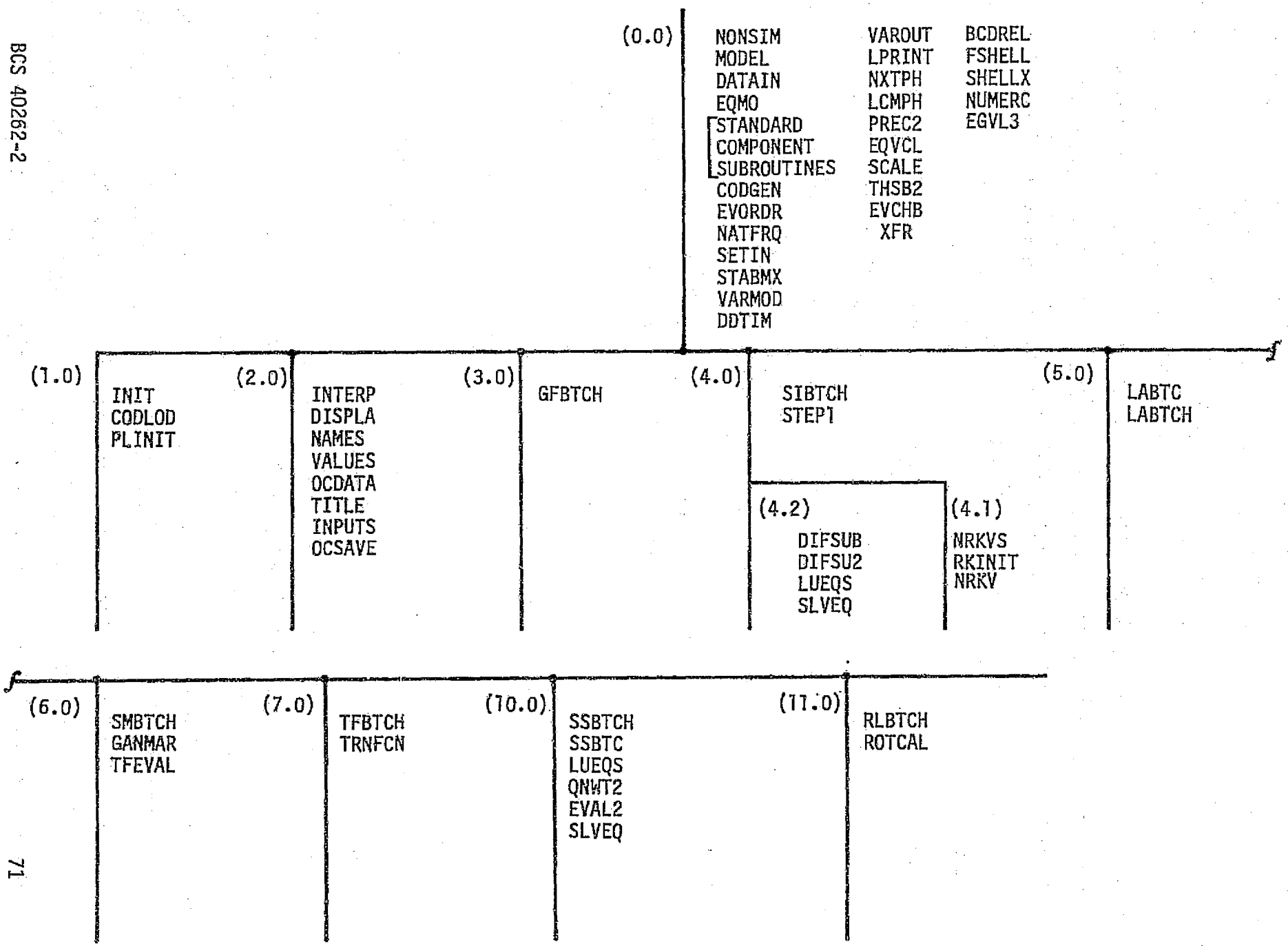

FIGURE 3.2-2 SIMWEST ANALYSIS PROGRAM - OVERLAY STRUCTURE

TABLE 3.2-1 OVERLAY DESCRIPTIONS

Overlay Level	Main Program Name	Description
(0,0)	NONSIM	Contains system model and frequently used routines such as eigenvalue-calculation routine, EGVL3.
(1,0)	INIT	Program initialization
(2,0)	INTERP	Interprets data input and analysis request commands
(3,0)	GFBTCH	Algebraic function scan
(4,0)	SIBTCH	Nonlinear simulation
(4,1)	NRKV	Runge-Kutta integration
(4,2)	DIFSUB	Gear integration
(5,0)	LABTC	Linear Analysis and Eigenvalue Sensitivity
(6,0)	SMBTCH	Stability Margin
(7,0)	TFBTCH	Transfer Function
(10,0)	SSBTC	Steady State Calculation
(11,0)	RLBTCH	Root Locus Calculation

The INTERP program is brought into core to interpret each input data card. Those program commands that involve only data storage or modification are performed by INTERP or one of the other routines in the (2,0) overlay. When an analysis request command is encountered, INTERP returns control to the main program which calls in the appropriate analysis overlay.

3.2.2 Command Interpretation

Figure 3.2-3 contains a macro flow diagram of the Analysis program command interpretation process. Each input data card is read and printed to provide a record of the progress through the analysis requests. Phrases are identified on each card by the routine NXTPH. When a blank phrase is encountered, a new card is read. Each phrase is tested against the three types: command phrases, program names, and program values. If one of these types is recognized, the proper action is taken. If the phrase is not one of these types, a test is made for an outstanding task. An outstanding task consists of such multiphrase tasks as defining state names, inputting parameter values, specifying initial conditions, etc. If there is no outstanding task, the warning message "CAN'T INTERPRET xxxxx" is printed and the program goes on to the next phrase.

3.2.3 Temporary Files

Two temporary files TAPE25 and TAPE30 are used by the Analysis program. TAPE25 serves as a temporary buffer for steady-state and simulation plot data. The plot data for each report interval is stored on TAPE25 until all report intervals for the steady-state analyses or the simulation analysis have been completed. Upon completion of the steady-state or simulation analysis, information describing the number of plots, report intervals, and plot scales are placed on TAPE30 and the plot data itself is transferred from TAPE25 to TAPE30. For other analyses such as root locus or transfer functions, the plot data is placed directly on TAPE30 upon the completion of the analysis.

Upon completion of all analyses for a particular run, TAPE 0 is processed by a separate program (NSMPPT) to generate lineprinter plots.

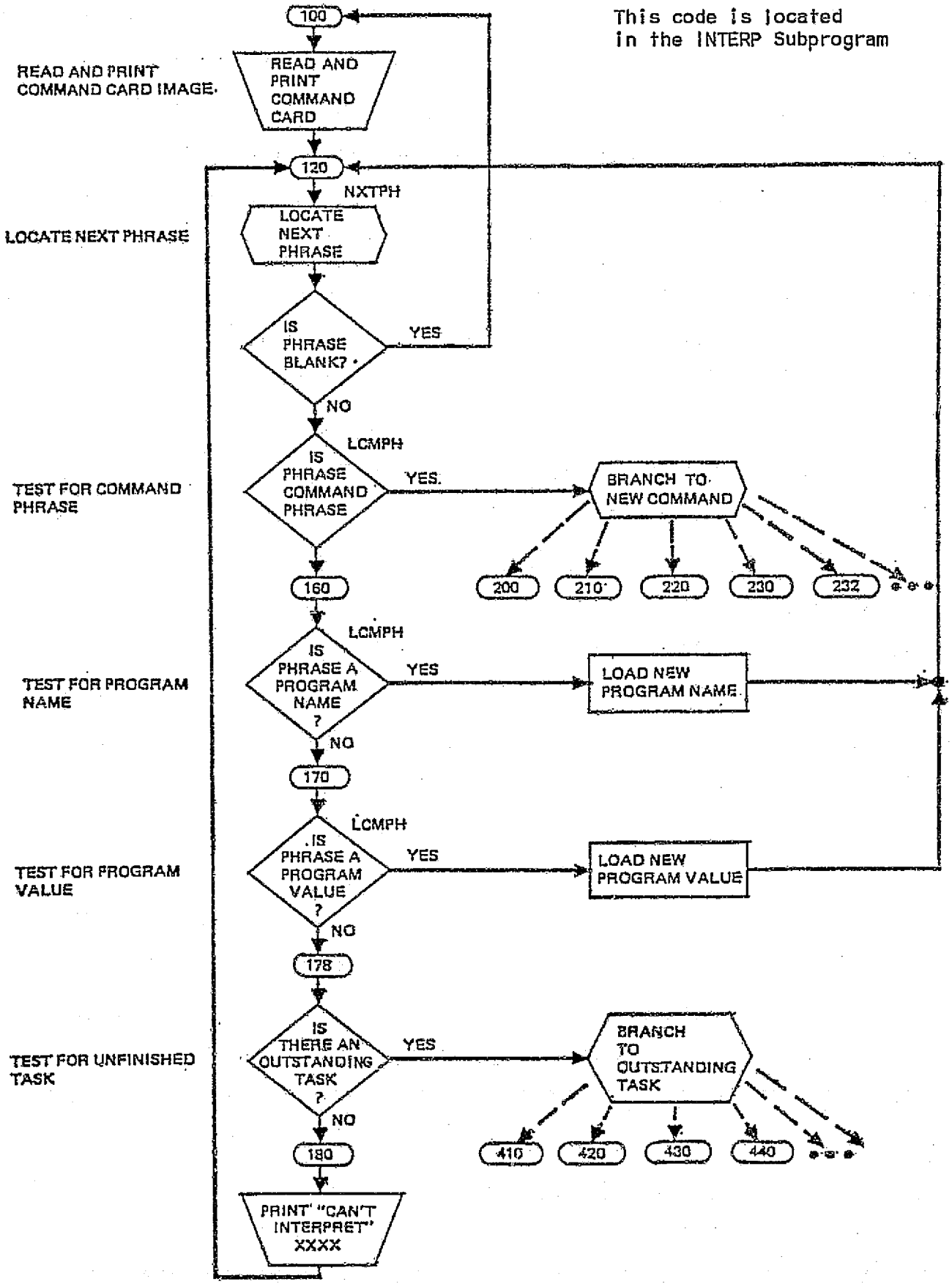

FIGURE 3.2-3 ANALYSIS PROGRAM COMMAND INTERPRETATION - MACRO FLOW DIAGRAM

3.3 ANALYSIS PROGRAM SOURCE LISTINGS

Compilation listings for the simulation program follows. Some subroutines such as NXTPH and LCMPH are used in several of the programs and will be found in the source listings for the FILOAD program (Section 4.3). There are five subroutines which are only called by the model EQMO or the library components. These are listed after the simulation program source. The names of the simulation routines, in order of appearance, are:

BLOCK DATA
CODGEN
CODLOD
DIFSUB
DIFSU2
DISPLA
DTTIM
EGVL3
EQVCL
EVAL2
EVCHB
EVORDR
FSHELL
GANMAR
GFBTCH
INIT
INPUTS
INTERP
LABTC
LABTCH
LPRINT
LUEQS
NAMES
NATFRQ
NONSIM
NRKV
NRKVS
PLINIT
PLOTAB
PREC2
QNWT2
RKINIT
RLBTCH
ROTCAL
SCALE
SETIN
SHELLX
SIBTCH
SLVEQ
SMBTCH
SSBTC
SSBTCH
STABMX
STEP1
TABIN
TFEVAL
TFBTCH
THSB2
TITLE
TRNFCN
VALUES
VARMOD
VAROUT
XFR

```
C BLOCKDA
      BLOCK DATA
C   VERSION 3.                              REVISED: APRIL 30 1976
      COMMON /CSIMUL/IPRIN,IPRATE,IOUT,NPTS,NPTMAX,INDMAX,TINC,TMAX
     1 ,INDEX,IPLOT,IDENT(4)
      COMMON/CPRON/PRONAM(8)/CPROV/PVALUE(27)/CSMPAR/SMPAR(10),ICIND(2)
      COMMON/CCOMM/ICOM(8),IPHRS,INDEXX
      COMMON /CSCALE/SCALE(5,4,6),NVAR(5 ,2,6),NPLTS(6)
      COMMON/CIO/IREAD,IWRITE,IDIAG
      COMMON/CPRINT/PRTNAM(10),LPRT(10)
      COMMON/COVRLY/INST,LOKSS,LOKSIM,CPUSEC
      REAL    IDENT,NVAR,PRONAM,SMPAR
      DATA IPRIN,IPRATE,IOUT,TINC,TMAX/0,1,1,-1,1./
      DATA IDENT/4*10HTIME      /
      DATA INDEX,IPLOT/0,1/,INDEXX/0/
      DATA PRONAM/8*10H          /,SMPAR/8*10H          /
      DATA PVALUE/-1.,1.,-1.,0.,1.,0.,1.,1.,3.,-1,1.,-1,100.,-1.,1.,5.,
     130.,0.,1.,6.,-10.,0.,0.,10.,0.,0.,0./
      DATA NPTMAX/1/,INDMAX/505/
      DATA NPLTS/3*1,3*0/,NVAR/30*10H           /,SCALE/120*0./
      DATA ICIND/2*0/
      DATA IREAD,IWRITE,IDIAG/5,6,0/
      DATA PRTNAM/10*10H          /,LPRT/10*0/
      DATA INST,LOKSS,LOKSIM/3*1/,CPUSEC/0./
      END
```

```
CCODGEN
      SUBROUTINE CODGEN(IDENT,IC,ICODE),RETURNS(R1)
C  PURPOSE:  GENERATE INTEGER IDENTIFICATION CODES GIVEN ALPHANUMERIC
C  CALL SEQUENCE:  IDENT = ALPHANUMERIC IDENTIFIER
C                  IC    = INITIAL CONDITION INDICATOR
C                  ICODE = INTEGER CODE NUMBER
C                  R1    = RETURN TAKEN WHEN IDENTIFIER CAN*T BE FOUND
C  CODE SCHEME:  THE SEVENTH COLUMN IS USED TO DESIGNATE WHICH GROUP
C                THE QUANTITY BELONGS.  THE FOLLOWING CODE IS USED:
C                  STATE VARIABLES      = 0
C                  STATE DERIVATIVES    = 1
C                  STATE I.C.*S         = 2
C                  VARIABLES            = 3
C                  PARAMETERS           = 4
C                              ICODE    = 0 IS USED FOR TIME
      COMMON/CNAMEX/NAMEX(1)/CNAMER/NAMER(1)/CNAMEV/NAMEV(1)/CNAMEP/
     1 NAMEP(1)
      COMMON/CORDER/NOX,NOV,NOP
      REAL IDENT,NAMEX,NAMER,NAMEV,NAMEP,NTIME,BLANK
      DATA NTIME/10HTIME      /,BLANK/10H          /
      IF(IDENT.EQ.BLANK)GO TO 260
C  TEST FOR TIME CODE
      IF(IDENT.NE.NTIME) GO TO 80
      ICODE=0
      RETURN
C  SEARCH STATE NAMELIST
80    CALL LCMPH(IDENT,NAMEX,NOX,1,ICODE)
      IF(ICODE.EQ.0) GO TO 90
      IF(IC.EQ.0) RETURN
      GO TO 255
C  SEARCH VARIABLES NAMELIST
90    CALL LCMPH(IDENT,NAMEV,NOV,1,ICODE)
      IF(ICODE.NE.0) GO TO 225
C  SEARCH RATES NAMELIST
      CALL LCMPH(IDENT,NAMER,NOX,1,ICODE)
      IF(ICODE.NE.0) GO TO 235
C  SEARCH PARAMETER NAMELIST
      CALL LCMPH(IDENT,NAMEP,NOP,1,ICODE)
      IF(ICODE.NE.0) GO TO 245
C  IDENTIFIER CAN*T BE RECOGNIZED.
      ICODE=-1
      RETURN R1
225   ICODE=ICODE+3000000
      RETURN
235   ICODE=ICODE+1000000
      RETURN
245   ICODE=ICODE+4000000
      RETURN
255   ICODE=ICODE+2000000
      RETURN
  260 ICODE=-1
      RETURN
      END
```

```
CCODLOD
      SUBROUTINE CODLOD(NAME,N,INITAL)
C  PURPOSE:  LOAD NAME ARRAYS WITH DEFAULT NAMES.
C  CALL SEQUENCE:  NAME    = N X 1 NAME ARRAY.
C                  N       = NO. OF NAMES IN ARRAY.
C                  INITAL  = INITIAL CHARACTER WORD.
      REAL NAME(N)
C   SCAN NAMES.
      DO 100 I=1,N
C  BLANK OUT NAME.
      NAME(I)=10H
C  PUT INITIAL CHARACTER IN 1ST CHARACTER OF NAME.
      CALL PUTT(NAME(I),1,INITAL)
C  CONVERT I TO BCD.
      ENCODE(10,11,NUM)I
11    FORMAT(I10)
      K=2
C  SCAN CHARACTERS OF NUM FOR NUMERIC VALUE.
      DO 50 J=1,10
C  GET JTH CHARACTER OF NUM.
      CALL GETT(NUM,J,KAR)
C  TEST FOR BLANK CHARACTERS AND SKIP THESE.
      IF(KAR.EQ.10H           ) GO TO 50
C  LOAD NON-BLANK CHARACTERS CONTAINING NUMERIC INTO NAME.
      CALL PUTT(NAME(I),K,KAR)
      K=K+1
   50 CONTINUE
  100 CONTINUE
      RETURN
      END
```

```
CDIFSUB
      OVERLAY(NONSIM,4,2)
      PROGRAM DIFSUB
C  PURPOSE:  PERFORM NUMERICAL INTEGRATION USING GEAR ALGORITHIM
C  VERSION 2.3                    REVISED: MARCH 23 1976
      COMMON/CTIME/TIME/CX/X(1)/ERMESS/IFATAL,IERR
      COMMON/CSINUL/DUM(6),TINC,TMAX
      COMMON/CORDER/NSIM,NOV,NOP
      COMMON/CWORK/W(1)
      COMMON /CDIFS/JSTART,KINIT,TP
C     IF FIRST CALL - INITIALIZE PARAMETERS
      IF(JSTART.NE.0) GO TO 9
      TIMD=TIME
      HMAX=TINC
      HMIN=AMIN1(1.E-5,TINC/10000.)
      H=HMIN*100.
      MAXDER=6
      EPS=1.E-05
      ND=NSIM
      M1=1
      M2=8*ND+1
      M3=M2+12*ND
      M4=M3+ND
      M5=M4+ND
      DO 5 I=1,NSIM
 5    W(M3-1+I)=AMAX1(1.,ABS(X(I)))
      CALL XFR(X,W,NSIM)
C     CHECK IF NEXT STEP WOULD INCREASE TIME PAST TP
9     TIME=TIMD
 10   CALL DIFSU2(NSIM,TIMD,W(M1),W(M2),H,HMIN,HMAX,EPS,W(M3),W(M4),
     1KINIT,JSTART,MAXDER,W(M5),ND)
      IF(KINIT.LE.0) GO TO 30
      IF(TIMD.GE.TP)GO TO 15
      GO TO 9
15    HE=(TP-TIMD)/H
      HEJ=HE
      CALL XFR(W,X,NSIM)
      DO 19 I=1,JSTART
      DO 17 J=1,NSIM
17    X(J)=X(J)+W(I*NSIM+J)*HEJ
19    HEJ=HEJ*HE
      TIME=TP
C ---------  TURN ON ERROR MESSAGES IN MODEL
      IERR=1
      CALL EQMO(TIME,H,0)
C ----------  TURN OFF ERROR MESSAGES IN MODEL
      IERR=0
      GO TO 100
30    CONTINUE
      WRITE(6,31)TIME,H,JSTART
31    FORMAT(//5X,*DROPPED BACK TO RUNGE-KUTTA METHOD FOR 100 STEPS AT:  ,
     1TIME=*,G13.5,* STEP SIZE=*,G13.5,* ORDER=*,I3)
      DO 32 I=1,NSIM
      J=M4-1+I
32    W(J)=W(J)/W(M3-1+I)
      WRITE(6,33)(W(M4-1+I),I=1,NSIM)
33    FORMAT(/50X,*RELATIVE ERRORS*/10(G13.5))
      WRITE(6,34)(W(M3-1+I),I=1,NSIM)
```

```
34     FORMAT(/30X,*MAX STATE,(MIN=1), SINCE LAST TINC INTERVAL*/
      1 10(G13.5))
       KINIT=0
       JSTART=0
       CALL XFR(W,X,NSIM)
 100   CONTINUE
       END
```

```
CDIFSU2
      SUBROUTINE DIFSU2(N,T,Y,SAVE,H,HMIN,HMAX,EPS,YMAX,ERROR,
     X                  KFLAG,JSTART,MAXDER,PW,ND)
C  VERSION 2.                          REVISED: JAN 7 1976
C N    THE NUMBER OF FIRST ORDER EQUATIONS
C      MAY BE DECREASED ON LATER CALLS IF NUMBER OF ACTIVE EQUATIONS REDU
C      BUT IT MUST NOT BE INCREASED WITHOUT CALLING WITH JSTART = 0.
C ND     FULL DIMENSION OF STATE VECTOR
C T    INDEPENDENT VARIABLE
C Y    8*N VECTOR  CONTAINING DEPENDENT VARIABLES AND SCALED DERIVATIVES
C      Y(I+(J-1)*ND) CONTAINS J-TH DERIVATIVE OF Y(I),SCALED BY H**J/J
C      ONLY Y(I) NEED BE PROVIDED BY CALLING PROGRAM ON FIRST ENTRY
C
C      IF INTERPOLATION TO NON MESH POINTS IS DESIRED AT T+E AND CURRENT
C      STEP SIZE IS H,LET S = E/H AND COMPUTE (YI)(T+E) = SUM Y(I+J*N)*S*
C                                              FROM J=0 TO J=NQ
C SAVE AT LEAST 12*N LOCATIONS
C H    STEP SIZE TO BE ATTEMPTED ON THE NEXT STEP.
C      IF VALUE PROVIDED BY THE USER DOES NOT CAUSE A LARGER ERROR THAN A
C      IT WILL BE USED. THE USER IS ADVISED TO USE A SMALL STEP FOR FIRST
C HMIN MIN STEP SIZE
C HMAX MAX STEP SIZE
C EPS  ERROR TEST CONSTANT. SINGLE STEP ERROR ESTIMATES DIVIDED BY YMAX(
C       MUST BE LESS THAN EPS IN EUCLIDEAN NORM.
C YMAX (N) ARRAY CONTAINING MAX SO FAR. NORMALLY SET TO 1 BEFORE FIRST E
C ERROR (N) ARRAY CONTAINING ESTIMATED ONE STEP ERROR
C KFLAG A COMPLETION CODE
C             +1 STEP WAS SUCCESFUL.
C             -1 STEP TAKEN WITH H = HMIN ,BUT REQUESTED ACCURACY NOT ACH
C             -2 MAXIMUM ORDER SPECIFIED WAS FOUND TO BE TOO LARGE
C             -3 CORRECTOR CONVERGENCE COULD NOT BE ACHIEVED FOR H.G7.HMI
C             -4 REQUESTED ERROR IS SMALLER THAN CAN BE HANDLED
C JSTART AN INPUT INDICATOR
C             -1 REPEAT THE LAST STEP WITH A NEW H
C              0 PERFORM THE FIRST STEP
C             +1 TAKE A NEW STEP CONTINUING FROM THE LAST
C        JSTART IS SET TO NQ, THE CURRENT ORDER OF THE METHOD AT EXIT.
C               NQ IS ALSO THE ORDER OF THE MAXIMUM DERIVATIVE AVAILABLE.
C         IT MUST BE LESS THAN 8 OR 7 FOR ADAMS OR STIFF METHODS RESPECTI
C MAXDER THE MAXIMUM DERIVATIVE THAT SHOULD BE USED
C PW     A BLOCK OF AT LEAST N**2 FLOATING POINT LOCATIONS.
      COMMON /CX/X(1)  /CXDOT/XDOT(1)
      DIMENSION Y(1),YMAX(1),SAVE(1),ERROR(1),PW(1),
     1          A(8),PERTST(7,3)
      DATA PERTST /2.0,4.5,7.333,10.42,13.7,17.15,1.0,
     1             3.0,6.0,9.167,12.5,15.98,1.0,1.0,
     1             1.,1.,0.5,0.1667,0.04133,0.008267,1.0/
      DATA A(2) / -1.0/
      IRET =1
      FPZ=1.E-15
      KFLAG = 1
      IF (JSTART.LE.0) GO TO 140
  100 DO 110 J=1,K
      JJ=(J-1)*ND
      DO 110 I = 1,N
  110 SAVE(JJ+I)=Y(JJ+I)
      HOLD = HNEW
      IF (H.EQ.HOLD) GO TO 130
```

```
 120 RACUM =H/HOLD
     IRET1 = 1
     GO TO 750
 130 NQOLD = NQ
     TOLD = T
     RACUM = 1.0
     IF (JSTART.GT.0) GO TO 250
     GO TO 170
 140 IF (JSTART.EQ.-1) GO TO 160
     NQ = 1
     N3 = ND
     N1 = N*10
     N2 = N1 + 1
     N5 = N1 + ND
     N8=8*ND
     N9=9*ND
     NE=ND**2+1
     N6 = N5 + 1
     CALL XFR(Y,X,N)
     CALL EQMO(T,H,0)
     DO 150 I = 1,N
 150 Y(ND+I) = XDOT(I)*H
     HNEW = H
     K = 2
     GO TO 100
 160 IF (NQ.EQ.NQOLD) JSTART = 1
     T = TOLD
     NQ = NQOLD
     K = NQ + 1
     GO TO 120
 170 IF (NQ.GT.6) GO TO 190
     GO TO (221,222,223,224,225,226),NQ
 190 KFLAG = -2
     RETURN
 221 A(1) = -1.
     GO TO 230
 222 A(1) =   6060 2525 2525 2525 2525B
     A(3) =   6061 2525 2525 2525 2525B
     GO TO 230
 223 A(1) =   6060 3505 6427 2135 0564B
     A(3) = A(1)
     A(4) =   6063 2135 0564 2721 3506B
     GO TO 230
 224 A(1) =   6061 0243 6560 5075 3412B
     A(3) =   6060 2314 6314 6314 6315B
     A(4) =   6062 1463 1463 1463 1463B
     A(5) =   6065 2702 4365 6050 7534B
     GO TO 230
 225 A(1) =   6061 0774 2064 2443 4016B
     A(3) =   6060 1334 3761 0321 2215B
     A(4) =   6061 3022 5372 3116 3664B
     A(5) =   6064 0774 2064 2443 4016B
     A(6) =   6070 0415 0510 7003 5713B
     GO TO 230
 226 A(1) =   6061 1360 2471 3602 4713B
     A(3) =   6060 0505 0505 0505 0505B
     A(4) =   6061 1252 5252 5252 5253B
     A(5) =   6063 1515 1515 1515 1515B
```

```
      A(6) =   6066 1717 1717 1717 1717B
      A(7) =   6072 3266 2155 1043 7732B
  230 K = NQ + 1
      IDOUB = K
      ENQ2 = .5/FLOAT(NQ + 1)
      ENQ3 = .5/FLOAT(NQ + 2)
      ENQ1 = 0.5/FLOAT(NQ)
      PEPSH = EPS
      EUP = (PERTST(NQ,2)*PEPSH)**2
      E = (PERTST(NQ,1)*PEPSH)**2
      EDWN =(PERTST(NQ,3)*PEPSH)**2
      IF (EDWN.EQ.0) GO TO 780
      BND = EPS*ENQ3/ FLOAT(N)
  240 IWEVAL = 2
      GO TO ( 250 , 680 ),IRET
  250 T = T + H
      DO 260 J = 2,K
        DO 260 J1 = J,K
          J2 = (K -J1 + J - 2)*ND
           J3=J2+ND
          DO 260 I = 1,N
  260   Y(J2+I)=Y(J2+I)+Y(J3+I)
      DO 270 I = 1,N
  270   ERROR(I) = 0.0
      DO 430 L = 1,3
      CALL XFR(Y,X,N)
      CALL EQMO(T,H,0)
      CALL XFR(XDOT,SAVE(N2),N)
      IF (IWEVAL.LT.1) GO TO 350
        GO TO 310
  290 N11 = N3 + 1
      N12 = N*N11 - N3
      DO 300 I = 1,N12,N11
  300   PW(I) = 1.0 + PW(I)
      IWEVAL = -1
      CALL LUEQS(PW,KK,KK,PW(NE),N,0,ND,1,1,FPZ,J1)
      IF(J1.EQ.0) GO TO 350
      GO TO 440
  310 DO 320 I = 1,N
  320   SAVE(N8+I) = Y(I)
      DO 340 J = 1,N
        R = EPS* AMAX1(EPS,ABS(SAVE(N8+J)))
        Y(J) = Y(J) + R
        D = A(1)*H/R
      N11=(J-1)*ND
      CALL XFR(Y,X,N)
      CALL EQMO(T,H,0)
      CALL XFR(XDOT,SAVE(N6),N)
        DO 330 I = 1,N
  330     PW(N11+I) = (SAVE(N5+I) - SAVE(N1+I))*D
  340   Y(J) = SAVE(N8+J)
      GO TO 290
  350 CONTINUE
  370   DO 380 I = 1,N
  380     SAVE(N5+I) = Y(ND+I) - SAVE(N1+I)*H
      CALL SLVEQ(PW,SAVE(N8+1),SAVE(N6),PW(NE),N,1,ND,N,N,FPZ,J1)
  410   NT = N
      DO 420 I = 1,N
```

```
      Y(I) = Y(I) + A(I)*SAVE(N8+I)
      Y(ND+I) = Y(ND+I) - SAVE(N8+I)
      ERROR(I) = ERROR(I) + SAVE(N8+I)
      IF  (ABS(SAVE(N8+I)).LE.(BND * YMAX(I))) NT = NT - 1
  420 CONTINUE
      IF (NT.LE.0) GO TO 490
  430 CONTINUE
  440 T = T - H
      IF ((H.LE.(HMIN*1.00001)).AND.((IWEVAL - 2).LT.-1)) GO TO 460
      IF (IWEVAL.NE.0) RACUM = RACUM *0.25
      IWEVAL = 2
      IRET1 = 2
      GO TO 750
  460 KFLAG = -3
  470 DO 480 J=1,K
      JJ=(J-1)*ND
      DO 480 I=1,N
  480     Y(JJ+I) = SAVE(JJ+I)
      H = HOLD
      NQ = NQOLD
      JSTART = NQ
      RETURN
  490 D = 0.0
      DO 500 I = 1,N
  500  D = D + (ERROR(I)/YMAX(I))**2
      IWEVAL = 0
      IF (D.GT.E) GO TO 540
      IF (K.LT.3) GO TO 520
      DO 510 J = 3,K
      JJ=(J-1)*ND
        DO 510 I = 1,N
  510     Y(JJ+I) = Y(JJ+I) + A(J)*ERROR(I)
  520 KFLAG = +1
      HNEW = H
      IF (IDOUB.LE.1) GO TO 550
      IDOUB = IDOUB - 1
      IF (IDOUB.GT.1) GO TO 700
      DO 530 I = 1,N
  530   SAVE(N9+I) = ERROR(I)
      GO TO 700
  540 KFLAG = KFLAG - 2
      IF (H.LE.(HMIN*1.00001)) GO TO 740
      T = TOLD
      IF (KFLAG.LE.-5) GO TO 720
  550 PR2 = (D/E)**ENQ2*1.2
      PR3 = 1.E+20
      IF ((NQ.GE.MAXDER).OR.(KFLAG.LE.-1)) GO TO 570
      D = 0.0
      DO 560 I = 1,N
  560   D = D + ((ERROR(I) - SAVE(N9+I))/YMAX(I))**2
      PR3 = (D/EUP)**ENQ3*1.4
  570 PR1 = 1.E+20
      IF (NQ.LE.1) GO TO 590
      D = 0.0
      JJ=(K-1)*ND
      DO 580 I = 1,N
  580   D = D + (Y(JJ+I)/YMAX(I))**2
      PR1 = (D/EDWN)**ENQ1*1.3
```

```
590 CONTINUE
    IF (PR2.LE.PR3) GO TO 650
    IF (PR3.LT.PR1) GO TO 660
600 R = 1.0/AMAX1(PR1,1.E-4)
    NEWQ = NQ - 1
610 IDOUB = 10
    IF ((KFLAG.EQ.1).AND.(R.LT.(1.1))) GO TO 700
    IF (NEWQ.LE.NQ) GO TO 630
    JJ=NEWQ*ND
    DO 620 I = 1,N
620    Y(JJ+I) = ERROR(I)*A(K)/FLOAT(K)
630 K=NEWQ+1
    IF (KFLAG.EQ.1) GO TO 670
    RACUM = RACUM*R
    IRET1 = 3
    GO TO 750
640 IF (NEWQ.EQ.NQ) GO TO 250
    NQ = NEWQ
    GO TO 170
650 IF (PR2.GT.PR1) GO TO 600
    NEWQ = NQ
    R = 1.0/AMAX1(PR2,1.E-4)
    GO TO 610
660 R = 1.0/AMAX1(PR3,1.E-4)
    NEWQ = NQ + 1
    GO TO 610
670 IRET = 2
    R = AMIN1(R,HMAX/ABS(H))
    H = H*R
    HNEW = H
    IF (NQ.EQ.NEWQ) GO TO 680
    NQ = NEWQ
    GO TO 170
680 R1 = 1.0
    DO 690 J = 2,K
      R1 = R1*R
    JJ=(J-1)*ND
      DO 690 I = 1,N
690     Y(JJ+I) = Y(JJ+I)*R1
    IDOUB = K
700 DO 710 I = 1,N
710   YMAX(I) = AMAX1(YMAX(I),ABS(Y(I)))
    JSTART = NQ
    RETURN
720 IF (NQ.EQ.1) GO TO 780
    CALL XFR(Y,X,N)
    CALL EQMO(T,H,0)
    CALL XFR(XDOT,SAVE(N2),N)
    R = H/HOLD
    DO 730 I = 1,N
      Y(I) = SAVE(I)
    SAVE(ND+I) = HOLD*SAVE(N1+I)
730 Y(ND+I) = SAVE(ND+I)*R
    NQ = 1
    KFLAG = 1
    GO TO 170
740 KFLAG = -1
    HNEW = H
```

```
      JSTART = NQ
      RETURN
  750 RACUM = AMAX1(ABS(HMIN/HOLD) ,RACUM)
      RACUM = AMIN1(RACUM,ABS(HMAX/HOLD))
      R1 = 1.0
      DO 760 J = 2,K
        R1 = R1*RACUM
        JJ=(J-1)*ND
        DO 760 I = 1,N
  760     Y(JJ+I) = SAVE(JJ+I)*R1
      H = HOLD*RACUM
      DO 770 I = 1,N
  770   Y(I) = SAVE(I)
      IDOUB = K
      GO TO (130 , 250 , 640 ),IRET1
  780 KFLAG = -4
      GO TO 470
      END
```

```
CDISPLA
      SUBROUTINE DISPLA(IDSPLY,IPHRS,MODE,ICOL)
C  PURPOSE:  INTERPRETS INPUT DATA PHRASES THAT DESCRIBE GRAPHIC DISPLAY
C  CALL SEQUENCE:  IDSPLY = DISPLAY NUMBER.
C                  IPHRS  = PHRASE TO BE INTERPRETED.
C                  MODE   = MODE = 1,2,3 INDICATES THAT VS,YRANGE,OR
C                           XRANGE RESPECTIVELY WAS THE LAST INTRUCTION.
C                  ICOL   = SET EQUAL TO THE COLUMN NUMBER IN SCALE.
      COMMON/CSCALE/SCALE(5,4,6),NVAR(5,2,6),NPLTS(6)
      REAL NVAR,IPHRS,LIST(3)
      DATA LIST/30HVS         YRANGE    XRANGE    /
C  CURRENT NUMBER OF PLOTS/DISPLAY.
      NPLT=NPLTS(IDSPLY)
C  SEARCH FOR COMMAND WORD.
      CALL LCMPH(IPHRS,LIST,3,1,ICODE)
      IF(ICODE.LE.0) GO TO 20
C  SAVE ICODE IN MODE AND BRANCH TO SET ICOL IF REQUIRED.
      MODE=ICODE
      GO TO (100,200,300),ICODE
C  TEST FOR NUMERIC PHRASE.
20    CALL NUMERC(IPHRS),RETURNS(60)
      IF(MODE.LE.2) GO TO 40
C  CONVERT X SCALE FROM A TO G FORMAT.
      CALL BCDREL(SCALE(NPLT,ICOL,IDSPLY),IPHRS)
      ICOL=4
      RETURN
C  CONVERT Y SCALE FROM A TO G FORMAT.
40    CALL BCDREL(SCALE(NPLT,ICOL,IDSPLY),IPHRS)
      ICOL=2
      RETURN
60    IF(MODE.EQ.1) GO TO 80
      NPLT=MIN0(NPLT+1,3)
      NPLTS(IDSPLY)=NPLT
C  LOAD Y AXIS NAME.
      NVAR(NPLT,1,IDSPLY)=IPHRS
      GO TO 90
C  LOAD X AXIS NAME.
80    NVAR(NPLT,2,IDSPLY)=IPHRS
90    MODE=-1
100   RETURN
C  SET COLUMN INDICATOR TO 1 FOR YRANGE.
200   ICOL=1
      RETURN
C  SET COLUMN INDICATOR TO 3 FOR XRANGE.
300   ICOL=3
      RETURN
      END
```

```
CDTTIM
      SUBROUTINE DTTIM (A)
C
C     GET THE CURRENT DATE AND TIME
C
      DIMENSION A(1)
      A(1) = DATE(I)
      A(2) = TIME(1)
      RETURN
      END
```

```
CEGVL3
      SUBROUTINE EGVL3(A,B,ER,EI,IA,IB,IC,ID,DW,FPZ,NA,MA)
C     SUBROUTINE TO COMPUTE EIGENVALUES OF A
C     INPUTS ARE;
C        A    THE SYSTEM MATRIX WHICH IS UNALTERED BY THIS PROGRAM
C        NA   THE ORDER OF THE SYSTEM
C        MA   THE ROW DIMENSION OF THE MATRIX A
C        FPZ  THE PRECISION INDICATOR
C
C     ON SUCCESSFUL COMPLETION (IERROR=0)
C       ER CONTAINS THE REAL PARTS OF THE EIGENVALUES
C       EI CONTAINS THE IMAGINARY PARTS OF THE EIGENVALUES
C
C     DIMENSION INFORMATION
C        B IS A NA**2 VECTOR
C        IA,IB,IC,ID,DW ARE NA LENGTH WORK VECTORS
C
C     THIS PROGRAM WAS DESIGNED AND CODED BY A. FREDERICK FATH OF
C     BOEING COMPUTER SERVICES, SEATTLE, WASHINGTON.  THIS VERSION
C     WAS COMPLETED DURING APRIL 1975.
C
      DIMENSION A(MA,1),ER(1),EI(1),B(1),IA(1),IB(1),IC(1),ID(1),DW(1)
      IERROR=0
      CALL PREC2(A,B,DW,IA,IB,IC,ID,NSM,NA,MA)
      CALL THSB2(B,IC,ID,NA,MA)
      CALL EVCHB(B,ER,EI,IC,FPZ,NSM,MA,IERROR)
      RETURN
      END
```

```
C$EQVCL
      SUBROUTINE EQVCL(N,NDIM,A,IPERM,NIX,ISTACK,IEQUIV,LOC)
C
C
C
C     PROGRAM TO DETECT AND ISOLATE EQUIVALENCE CLASSES UNDER
C     REACHABILITY WITHIN A GRAPH, GIVEN A CONNECTION MATRIX OF
C     THE GRAPH.  THE OUTPUT IS AVAILABLE IN STACK AND EQUIV.
C
C
C     DESIGNED BY E. MCCREIGHT, NOVEMBER, 1969.
C        INFORMATION SCIENCES LABORATORY
C        BOEING SCIENTIFIC RESEARCH LABORATORIES
C        SEATTLE, WASHINGTON
C
      DIMENSION A(NDIM,NDIM),ISTACK(NDIM),LOC(NDIM),IEQUIV(NDIM)
      DIMENSION IPERM(NDIM)
C
C     INITIALIZE THE VECTORS WHICH INDICATE THE EQUIVALENCES DISCOVERED
C     AND THE ROWS COMPLETELY OR PARTIALLY PROCESSED.
C
      ISTKP=1
      IEQCP=N
      DO 8003 I=1,N
 8003 LOC(I)=0
C
C     TRY TO MAKE EACH ROW IN SUCCESSION THE ROOT OF A DEPENDENCY TREE.
C
      I=1
 8000 CONTINUE
C
C     HAS THIS ROW ALREADY BEEN PROCESSED
      IF (LOC(I).EQ.-1) GO TO 8100
C
C
C     START AT THE LEFT OF THE ROW AND WORK TO THE RIGHT.
C
 8004 J=1
      LOC(I)=ISTKP
      IEQUIV(I)=I
C
C     READ ACROSS THE ROW. WHEN YOU COME TO A NON-ZERO ENTRY, BREAK
C     OUT.
C
 8005 IX1=IPERM(I)
      DO 8010 K=J,N
      IF (A(IX1,IPERM(K)).EQ.0.D0) GO TO 8010
C
C
C     DID WE DISCOVER I TO BE EQUIVALENT TO ITSELF  WE ALREADY KNEW
C     THAT.
C
 8020 IF (K.EQ.I) GO TO 8010
C
C     DID WE DISCOVER I TO POINT TO SOME ROW WHICH WE HAVE ALREADY AT
C     LEAST PARTIALLY PROCESSED   THIS IMPLIES THAT I IS EQUIVALENT TO
C     THE LOWEST ROW TO WHICH THAT ROW IS EQUIVALENT, IF THAT ROW IS
C     STILL IN THE STACK.
```

```
C
      IF (LOC(K).NE.0) GO TO 8050
C
C     DID WE DISCOVER I TO POINT TO SOME ROW WHICH HAS NEVER BEEN
C     UNDER CONSIDERATION  IF SO, INTERRUPT EVERYTHING AND CONSIDER IT
C     NOW.
C
      ISTACK(ISTKP)=I
      ISTKP=ISTKP+1
      I=K
      GO TO 8004
C
C     FIND THE LOWEST ROW TO WHICH ROW K IS EQUIVALENT. IF THIS LOWEST
C     ROW IS STILL IN THE STACK, SEE IF IT IS THE LOWEST ROW IN THE ST-
C     ACK TO WHICH WE KNOW ROW I TO BE EQUIVALENT.
C
 8050 IF (LOC(K).GT.0) GO TO 8051
      K1=IEQUIV(K)
      IF (LOC(K1).LE.0) GO TO 8010
      GO TO 8052
 8051 K1=K
 8052 IF (LOC(K1).GE.LOC(IEQUIV(I))) GO TO 8010
      IEQUIV(I)=K1
 8010 CONTINUE
C
C     THE READ ACROSS ROW I IS COMPLETE. INDICATE THIS.
C
 8015 LOC(I)=-1
C
C     TRY TO MOVE BACK TOWARD THE ROOT ROW. IF THIS IS THE ROOT ROW,
C     THEN SELECT A NEW ROOT ROW AND MOVE THIS ROW TO THE OUTPUT QUEUE.
C
 8014 IF (ISTKP.EQ.1) GO TO 8090
      ISTKP=ISTKP-1
C
C
C     SET UP ROW I WITH ITS NEW EQUIVALENCE AND TEST WHETHER THE EQUIV-
C     ALENCE IS TRIVIAL.
C
 8016 K2=IEQUIV(I)
 8019 IRV=1
      IF (IEQUIV(I).EQ.I) GO TO 8091
C
C
C     SEE IF THE BEST EQUIVALENCE NOW KNOWN FOR ROW I IS IN FACT BETTER
C     THAN THE BEST KNOWN EQUIVALENCE FOR I*S FATHER AS WELL. IF SO,
C     RECORD IT IN THE VECTOR LOWST.
C
      K1=ISTACK(ISTKP)
      IF (LOC(IEQUIV(K1)).LT.LOC(K2)) GO TO 8018
      IEQUIV(K1)=K2
 8018 CONTINUE
C
C     NOW COMPUTE THE TRANSITIVE CLOSURE OF THIS NEW EQUIVALENCE CLASS.
C
      DO 8017 K1=1,N
      IF (IEQUIV(K1).EQ.I) IEQUIV(K1)=K2
 8017 CONTINUE
```

```
C
C       BACK UP TOWARD THE ROOT, USING THE STACK TO GUIDE US.
C
 8024 J=I+1
      I=ISTACK(ISTKP)
      IF (J.GT.N) GO TO 8015
      GO TO 8005
C
C
C       ENTER THE CANONICAL ELEMENT OF AN EQUIVALENCE CLASS INTO THE
C       QUEUE OF SUCH ELEMENTS. IF THE QUEUE IS EMPTY, INITIALIZE IT.
C       OTHERWISE ENTER OUR NEW ELEMENT AT THE TAIL.
C
 8090 IRV=2
 8091 ISTACK(IEQCP)=I
      IEQCP=IEQCP-1
      GO TO (8024,8100),IRV
C
C       END OF MAIN LOOP.
C
 8100 IF(I.GE.N) GO TO 8104
      I=I+1
      GO TO 8000
 8104 CONTINUE
C
C       NOW TRANSFORM THE EQUIVALENCE CLASSES TO PATH NORMAL FORM:
C       CONTIGUOUS PARTITION BLOCKS IN OUT1 WITH THEIR SIZES IN OUT2.
C
C       FIRST FORM A CHAIN FOR EACH EQUIVALENCE CLASS.
C
      DO 8105 I=1,N
 8105 LOC(I)=0
      DO 8110 I=1,N
      J=IEQUIV(I)
      IF (J.EQ.I) GO TO 8110
      LOC(I)=LOC(J)
      LOC(J)=I
 8110 CONTINUE
C
C       NOW PROCEED THROUGH THE EQUIVALENCE CLASSES LISTED IN THE QUEUE,
C       ENUMERATING EACH EQUIVALENCE CLASS INTO THE QUEUE STACK, AND
C       COUNTING EACH CLASS INTO QUEUE EQUIV.
C
      IOIX=1
      NIX=1
 8111 IF (IEQCP.EQ.N) GO TO 8120
      IEQCP=IEQCP+1
      I=ISTACK(IEQCP)
      J=LOC(I)
      ISTACK(IOIX)=I
      IEQUIV(NIX)=1
 8115 IOIX=IOIX+1
      IF (J.EQ.0) GO TO 8119
      ISTACK(IOIX)=J
      IEQUIV(NIX)=IEQUIV(NIX)+1
      J=LOC(J)
      GO TO 8115
 8119 NIX=NIX+1
```

```
      GO TO 8111
8120  CONTINUE
      NIX=NIX-1
      RETURN
      END
```

```
CEVAL2
      SUBROUTINE EVAL2(XT,N,FUN,P,RMS,F)
C   VERSION 3.                              REVISED: JUNE 4 1976
C   PURPOSE: EVALUATE MODEL RATES AND CALCULATE THEIR RMS VALUE
C   CALL SEQUENCE:  XT    - STATE VECTOR
C                   N     - TOTAL NUMBER OF STATES
C                   FUN   - FUNCTION TO EVALUATE RATES  (EQMO)
C                   P     - (NOT USED)
C                   RMS   - RMS VALUE OF RATES
C                   F     - RATES  (RESIDULES)
      COMMON /CX/X(1)/CXDOT/XDOT(1)/CINT/INT(1)/CTIME/TIME
      DIMENSION XT(1),F(1)
      J=0
      DO 100 I=1,N
      IF(INT(I).EQ.0)GO TO 100
      J=J+1
      X(I)=XT(J)
100   CONTINUE
      CALL FUN(TIME,TIME,1)
      RMS = 0.
      J=0
      DO 110 I=1,N
      IF(INT(I).EQ.0)GO TO 110
      J=J+1
      F(J)=XDOT(I)
      RMS = RMS+F(J)*F(J)
110   CONTINUE
      RMS = SQRT(RMS)
      RETURN
      END
```

```
CEVCHB
      SUBROUTINE EVCHB(A,EVR,EVI,IC,FPZ,NSM,MM,IERROR)
C     SUBROUTINE TO CALCULATE EIGENVALUES OF MATRIX A
C     AND RETURN THE REAL PARTS IN EVR AND THE IMAGINARY PARTS IN EVI.
C     IC IS THE BLOCKING INFORMATION VECTOR INDICATING THE IRREDUCIBLE
C     BLOCKS CONTAINING THE EIGENVECTORS OF A.
C     NSM IS THE NUMBER OF SUCH BLOCKS.
C     MM IS THE ROW DIMENSION OF A
C     QR ALGORITHM FROM COMPUTER J., VOL. 11, NUM.1,
C     MAY 1968, PP. 112-114,  ALGORITHM 32 BY GRAD,REDISH,BREBNER
C     MODIFIED TO PREVENT SHIFT CYCLING.
C     FPZ IS FINITE PRECISION ZERO
C
C     THIS PROGRAM WAS DESIGNED AND CODED BY A. FREDERICK FATH OF
C     BOEING COMPUTER SERVICES, SEATTLE, WASHINGTON.  THIS VERSION
C     WAS COMPLETED DURING APRIL 1975.
C
      DIMENSION A(MM,1),EVR(1),EVI(1),IC(1)
      JT=1
      ICOUNT =0
      IERROR=0
      DO 350 IT=1,NSM
      IF(IT.EQ.1) GO TO 320
      JT=IC(IT-1)+1
320   KT=IC(IT)-JT+1
      IF(KT.NE.1) GO TO 340
      EVR(JT)=A(JT,JT)
      EVI(JT)=0.
      GO TO 350
340   IA=JT
      NA=KT
9     SHIFT=0.
      N=IA+NA-1
      MAXST=NA*10
      IF(A(N,N).NE.0.) GO TO 1
      IF(NA.LE.2) GO TO 1
      IF(A(N-1,N-1).NE.0.) GO TO 1
      IF(A(N-1,N).NE.0.) GO TO 1
      SHIFT=A(N,N-1)
1     X=0.
      DO 5 K=IA,N
      IF(K.NE.IA) GO TO 2
      M=IA
      GO TO 3
2     M=K-1
3     DO 5 I=M,N
5     X=X+A(K,I)**2
      E=SQRT(X)
      ARB=E/(N-IA)
      E=E*FPZ
      M=N
      NS=0
      NSOLD=0
151   IP=1
      IF(M-1.GE.1)OLD1=ABS(A(M,M-1))+1.
      IF(M-2.GE.1) OLD2=ABS(A(M-1,M-2))+1.
10    K=M-1
      M1=K
```

```
      I=K
      IF(K-IA+1) 99,11,12
12    IF(M-2.EQ.0) GO TO 13
      IF(ABS(A(M,K)).LE.E) GO TO 11
16    I=I-1
      IF(ABS(A(K,I)).LE.E) GO TO 17
      K=I
      IF(K.GT.IA) GO TO 16
17    IF(K.EQ.M1) GO TO 13
      IF(IP.EQ.2) GO TO 153
      IF(ABS(A(M,M1)).LT.OLD1) GO TO 155
      IP=2
153   IF(ABS(A(M1,M1-1)).LT.OLD2) GO TO 154
      IP=1
      IF(NS.LT.NSOLD+4) GO TO 157
      NSOLD=NS
      S=ARB
      R=0.
      GO TO 156
155   OLD1=ABS(A(M,M1))
154   OLD2=ABS(A(M1,M1-1))
157   CONTINUE
      S=A(M,M)+A(M1,M1)+SHIFT
      R=A(M,M)*A(M1,M1)-A(M,M1)*A(M1,M)+SHIFT**2*.25
156   A(K+2,K)=0.
      I=K+1
      X=A(K,K)*(A(K,K)-S)+R+A(K,I)*A(I,K)
      Y=A(I,K)*(A(K,K)+A(I,I)-S)
      Z=A(K+2,I)*A(I,K)
      SHIFT=0.
      NS=NS+1
      ICOUNT=ICOUNT + (M-K)**2
      DO 29 I=K,M1
      I1=I+1
      I2=I+2
      I3=I+3
      IF(I.EQ.K) GO TO 18
      X=A(I,I-1)
      Y=A(I1,I-1)
      IF(I2.LE.M) GO TO 19
      Z=0.
      GO TO 18
19    Z=A(I2,I-1)
18    S=SQRT(X**2+Y**2+Z**2)
      SR=S
      IF(X.LT.0.) GO TO 20
      S=-S
20    IF(I.EQ.K) GO TO 21
      A(I,I-1)=S
21    IF(SR.GE.E*1.E-05) GO TO 30
      IF(I3.GT.M) GO TO 29
      GO TO 28
30    AL=1.-X/S
      S=X-S
      X=Y/S
      Y=Z/S
      DO 23 J=I,M
      S=A(I,J)+A(I1,J)*X
```

```
      .
      IF(I2.GT.M) GO TO 22
      S=S+A(I2,J)*Y
22    S=S*AL
      A(I,J)=A(I,J)-S
      A(I1,J)=A(I1,J)-S*X
      IF(I2.GT.M) GO TO 23
      A(I2,J)=A(I2,J)-S*Y
23    CONTINUE
      L=I2
      IF(I.LT.M1) GO TO 24
      L=M
24    DO 26 J=K,L
      S=A(J,I)+A(J,I1)*X
      IF(I2.GT.M) GO TO 25
      S=S+A(J,I2)*Y
25    S=S*AL
      A(J,I)=A(J,I)-S
      A(J,I1)=A(J,I1)-S*X
      IF(I2.GT.M) GO TO 26
      A(J,I2)=A(J,I2)-S*Y
26    CONTINUE
      IF(I3.GT.M) GO TO 29
      S=-A(I3,I2)*Y*AL
28    A(I3,I)=S
      A(I3,I1)=S*X
      A(I3,I2)=S*Y+A(I3,I2)
29    CONTINUE
      IF(NS.GT.MAXST) GO TO 6
      GO TO 10
11    EVR(M)=A(M,M)
      EVI(M)=0.
      M=K
      GO TO 151
13    R=(A(K,K)+A(M,M))/2.
      S=(A(M,M)-A(K,K))/2.
      S=S*S+A(K,M)*A(M,K)
      IF(S.LT.0.) GO TO 14
      S=SQRT(S)
      EVR(K)=R-S
      EVR(M)=R+S
      EVI(K)=0.
      EVI(M)=0.
15    M=M-2
      GO TO 151
14    S=SQRT(-S)
      EVR(K)=R
      EVR(M)=R
      EVI(K)=S
      EVI(M)=-S
      GO TO 15
6     WRITE(6,7) MAXST
7     FORMAT(*0  NO CONVERGENCE AFTER NUMBER OF QR ITERATIONS =*,I6)
      IERROR=1
99    CONTINUE
350   CONTINUE
      RETURN
      END
```

```
CEVORDR
      SUBROUTINE EVORDR(EVR,EVI,ID,NLIN)
C  PURPOSE:  ORDER EIGENVALUES TO HAVE INCREASING NEGATIVE
C            REAL PARTS.
C  CALL SEQUENCE:  EVR  - NLIN X 1 ARRAY REAL PARTS OF EIGENVALUES
C                  EVI  - NLIN X 1 ARRAY IMAG. PARTS OF EIGENVALUES
C                  ID   - NLIN X 1 WORK ARRAY
C                  NLIN - SYSTEM ORDER
C  DESIGNED BY: J.D. BURROUGHS              FEB 1974
      DIMENSION EVR(1),EVI(1),ID(1)
C                     *** ORDER EIGENVALUES ***
      CALL FSHELL(EVR,ID,NLIN)
      CALL SHELLX(EVI,ID,NLIN)
C                  *** REVERSE EIGENVALUE ORDER ***
      NLIN1=NLIN+1
      NLIN2=NLIN/2
      DO 100 I=1,NLIN2
      I2=NLIN1-I
      EVRS=EVR(I)
      EVIS=EVI(I)
      EVR(I)=EVR(I2)
      EVI(I)=EVI(I2)
      EVR(I2)=EVRS
  100 EVI(I2)=EVIS
      I=1
  120 IF(EVI(I)) 160,180,140
  140 I=I+2
      GO TO 200
  160 EVI(I)=ABS(EVI(I))
      I=I+1
      EVI(I)=-ABS(EVI(I))
  180 I=I+1
  200 IF(I.LT.NLIN) GO TO 120
      RETURN
      END
```

```
CFSHELL
      SUBROUTINE FSHELL (IARRAY,KEY,N)
C  PURPOSE:  ORDER AN ARRAY TO HAVE INCREASING MAGNITUDE AND
C            FORM KEY FOR ORDERING RELATED ARRAY.
C  CALL SEQUENCE:  IARRAY - N X 1 ARRAY OF VALUES TO BE SORTED
C                  KEY    - N X 1 ARRAY OF KEYS FOR SORTING DEPENDENT
C                           ARRAY
C                  N      - NUMBER OF ELEMENTS TO BE SORTED.
      DIMENSION IARRAY(1),KEY(1)
      DO 10 I=1,N
   10 KEY(I)=I
      M=N
   20 M=M/2
      IF(M)30,30,40
   30 RETURN
   40 K=N-M
      DO 70 J=1,K
      I=J
   50 II=I+M
      IF(IARRAY(I)-IARRAY(II))70,70,60
   60 LIMBO=IARRAY(I)
      IARRAY(I)=IARRAY(II)
      IARRAY(II)=LIMBO
      LIMBO=KEY(I)
      KEY(I)=KEY(II)
      KEY(II)=LIMBO
      I=I-M
      IF(I)70,70,50
   70 CONTINUE
      GO TO 20
      END
```

```
CGANMAR
      SUBROUTINE GANMAR(NSIM,IACT,PARA,KMAX,IPOLE,GMDSPY,A,RATIO,
     1 DWORK,IA,IB,IC,ID,POLE,EVR,EVI,XDOTO,POLES),RETURNS(R1)
C  VERSION 2.                          REVISED: DEC 23 1975
C  PURPOSE:  CALCULATE STABILITY MARGINS OF ONE OR MORE MODEL PARAMETERS
C  CALL SEQUENCE:  NSIM  - MODEL ORDER
C                  IACT  - ARRAY OF SM PARAMETERS (IDENTIFICATION CODES)
C                   PARA  - ARRAY OF SM PARAMETERS  (HOLLORITH NAMES)
C                   KMAX  - NUMBER OF SM PARAMETERS
C                   IPOLE  - SPECIFIES IF STABILITY MATRIX MUST BE CALC.
C                                (IPOLE = 0 == CALC.)
C                   GMDSPY - ARRAY CONTAINING SM ANALYSIS RESULTS
C               NAME                DESCRIPTION          LOCATION
C                  A       - NSIM X NSIM  WORK ARRAY    /CWORK/A(1)
C                  RATIO   - NSIM X NSIM  WORK ARRAY    /CWORK/A(NN)
C                  DWORK   - NSIM X 1  WORK ARRAY       /CWORK/A(N1)
C                  IA      - NSIM X 1  WORK ARRAY       /CWORK/A(N2)
C                  IB      - NSIM X 1  WORK ARRAY       /CWORK/A(N3)
C                  IC      - NSIM X 1  WORK ARRAY       /CWORK/A(N4)
C                  ID     - NSIM X 1 WORK ARRAY        /CWORK/A(N5)
C                  POLE    - NSIM X 1  WORK ARRAY       /CWORK/A(N6)
C                  EVR     - NSIM X 1  WORK ARRAY       /CWORK/A(N2)
C                  EVI     - NSIM X 1  WORK ARRAY       /CWORK/A(N3)
C                  XDOTO   - NSIM X 1  WORK ARRAY       /CWORK/A(N1)
C                  POLES   - NSIM X 1  WORK ARRAY       /CWORK/A(N4)
C               RETURN R1 -- RETURN TAKEN IF NOMINAL SYSTEM IS UNSTABLE
C  DESIGNED BY: J.D. BURROUGHS                    JAN 1969
      REAL EVR(1),EVI(1),GMDSPY(1),RMAG(20),FREQ(20),XDOTO(1)
      COMPLEX POLES(1),POLE(1)
      COMPLEX OMEGA,OMEGA1,OMEGA2,OMEGA3,R
      DIMENSION IACT(10),A(1),RATIO(1)
      DIMENSION DWORK(1),IA(1),IB(1),IC(1),ID(1)
      REAL PARA(10)
      DATA IPOM/1OH+-+-+-+-+-/,IBLNK/1OH          /
      INDEXF(I1,I2,M1)=I1+(I2-1)*M1
      IF(IPOLE.GT.0) GO TO 29
C ========  FORM STABILITY MATRIX AND CALC. EIGENVALUES
      CALL STABMX(NSIM,XDOTO,ICOUNT,RATIO,A,N,0)
      CALL EGVL3(A,RATIO,EVR,EVI,IA,IB,IC,ID,DWORK,1.E-14,N,N)
      DO 10 I=1,N
   10 POLE(I)=CMPLX(EVR(I),EVI(I))
      WRITE(6,21)
   21 FORMAT(1HO/27X,20HNOMINAL SYSTEM POLES)
C ==========  CALC. NATURAL FREQUENCIES AND DAMPING RATIOS
      CALL NATFRQ(EVR,EVI,RATIO,DWORK,N,NPOLES)
      WRITE(6,2867) N
 2867 FORMAT(     28X,I3,2X,*EIGENVALUES*/13X,*REAL*,9X,*IMAGINARY*,
     1 6X,*NATURAL FREQ.*,5X,*DAMPING RATIO*)
      DO 2868 I=1,NPOLES
      J=IBLNK
      IF(EVI(I).GT.0.) J=IPOM
 2868 WRITE(6,2869)I,EVR(I),J,EVI(I),RATIO(I),DWORK(I)
 2869 FORMAT(3X,I3,3X,G12.6,2X,A2,G12.6,4X,2G16.6)
      IPOLE=2
      FMAX=0.
C =======  DETERMINE MAXIMUM NATURAL FREQUENCY OF SYSTEM
      DO 25 I=1,N
      POMAG=CABS(POLE(I))
```

```
      IF(POMAG.GT.FMAX) FMAX=POMAG
   25 CONTINUE
C ========  SET FMAX = TWICE MAX. NAT. FREQ. OF MODEL
C            (THIS LIMITS RANGE OF SEARCH FOR ZERO PHASE)
      FMAX=2.*FMAX
C ==========  TEST FOR UNSTABLE SYSTEM
      DO 41 I=1,N
      IF(EVR(I).GE.0.) GO TO 28
   41 CONTINUE
      GO TO 29
   28 CONTINUE
      WRITE(6,27)
   27 FORMAT(1HO,20X,40H***WARNING*** NOMINAL SYSTEM IS UNSTABLE)
      RETURN R1
C =========  START STABILITY MARGIN ANALYSIS ================
   29 K=0
C ========  ASSUME DIVERGENT WITH SM PARAMETER = 0
   30 IZERO=0
      K=K+1
      CALL VAROUT(IACT(K),P)
      GMDSPY(INDEXF(K,1,KMAX))=P
C ==========  SKIP ANALYSIS FOR SM PARAMETERS WITH 0 NOMINAL VALUES
      IF(P.EQ.0.) GO TO 2010
      WRITE(6,2011) PARA(K),P
 2011 FORMAT(1HO/20X,*NOMINAL VALUE OF PARAMETER *,A8,3H = ,G12.6//)
C ========  SAVE NOMINAL VALUE OF SM PARAMETER
      GAINO=P
C ==========  SET SM PARAMETER = 0
      CALL VARMOD(IACT(K),0.)
C =========  CALC. STABILITY MATRIX AND EIGENVALUES
      CALL STABMX(NSIM,XDOTO,ICOUNT,RATIO,A,N,0)
      CALL EGVL3(A,RATIO,EVR,EVI,IA,IB,IC,ID,DWORK,1.E-14,N,N)
      DO 165 I=1,N
  165 POLES(I)=CMPLX(EVR(I),EVI(I))
C ========  CHECK SYSTEM STABILITY WITH SM PARAMETER = 0
  125 DO 170 I=1,N
      IF(EVR(I).GE.0.) GO TO 175
  170 CONTINUE
      WRITE(6,171)   PARA(K)
  171 FORMAT(1HO,40X,*THE SYSTEM IS STABLE WITH *,A8,5H = 0.)
C ========  SET LOWER STABILITY BOUND = 0 AND FREQ = 1111
C              (DEFAULT VALUE WHEN STABLE)
      GMDSPY(INDEXF(K,2,KMAX))=0.
      GMDSPY(INDEXF(K,3,KMAX))=1111.
C =========  STABLE WITH SM PARAMETER = 0
      IZERO=1
      GO TO 180
C ========  TEST FOR POLE ON IMAGINARY AXIS WHEN SM PARAMETER = 0
175   IF(EVR(I).GT.0)GO TO 180
C ========  POLE ON IMAGINARY AXIS WHEN SM PARAMETER = 0
C ========  LOAD SM PARAMETER VALUE AND FREQUENCY
      RMAG(1)=0.
      FREQ(1)=EVI(I)
      INDEX=1
C ========  POLE ON IMAGINARY AXIS WITH SM PARAMETER =0
      IZERO=-1
  180 WRITE(6,2030) PARA(K)
 2030 FORMAT(30X,*POLES WITH *,A8,5H = 0.)
```

```
C =========  CALC. NATURAL FREQUENCIES AND DAMPING RATIOS
      CALL NATFRQ(EVR,EVI,RATIO,DWORK,N,NPOLES)
      WRITE(6,2867) N
C =========  PRINT EIGENVALUES WITH SM PARAMETERS = 0
      DO 2870 I=1,NPOLES
      J=IBLNK
      IF(EVI(I).GT.0.) J=IPOM
 2870 WRITE(6,2869)I,EVR(I),J,EVI(I),RATIO(I),DWORK(I)
      IF(IZERO.EQ.-1)GO TO 210
      INDEX=0
      OMEGA2=(0.,0.)
C ========  EVALUATE TRANSFER FUNCTION MAGNITUDE AND PHASE AT 0 FREQ.
      CALL TFEVAL(OMEGA2,POLES,POLE,N,R,LFLAG,IQUAD2,PHASE2)
      IF(REAL(R)) 210,200,200
C ========  REAL DIVERGENCE INDICATED
  200 INDEX=1
      RMAG(1)=1./CABS(R)
      FREQ(I)=0.
  210 OMEGA=(0.,.01)
      OMEGA2=OMEGA
C ========  EVALUATE TRANSFER FUNCTION AT .01 FREQ.
      CALL TFEVAL(OMEGA2,POLES,POLE,N,R,LFLAG,IQUAD2,PHASE2)
C          SEARCH FOR PHASE ANGLE QUADRANT TRANSITION
C              GEOMETRIC SEARCH TECHNIQUE
  220 OMEGA=1.2*OMEGA
C ========  END SEARCH AT 2*MAX. NOMINAL NATURAL FREQ.
      IF(AIMAG(OMEGA).GT.FMAX) GO TO 400
  215 CONTINUE
      CALL TFEVAL(OMEGA,POLES,POLE,N,R,LFLAG,IQUAD,PHASE)
      IF(IABS(IQUAD-IQUAD2)-2) 230,225,300
C =======  CHANGED MORE THAN 1 QUADRANT, REDUCE STEP SIZE AND
C             CONTINUE GEOMETRIC SEARCH
  225 OMEGA=.91667*OMEGA
      GO TO 215
C ========  CONTINUE GEOMETRIC SEARCH
  230 PHASE2=PHASE
      OMEGA2=OMEGA
      IQUAD2=IQUAD
      GO TO 220
C ========  ZERO CROSSING OCCURED,  START DICHOTOMOUS SEARCH
  300 OMEGA1=OMEGA
      PHASE1=PHASE
      IQUAD1=IQUAD
C ==========  DICHOTOMOUS SEARCH FOR ZERO PHASE
      DO 340 I=1,50
      OMEGA3=.5*(OMEGA1+OMEGA2)
      CALL TFEVAL(OMEGA3,POLES,POLE,N,R,LFLAG,IQUAD3,PHASE3)
C =========  TEST FOR CONVERGENCE
      IF(LFLAG) 320,320,310
C =========  CONVERGENCE OCCURED, SAVE STABILITY MARGIN AND OSCILLATION
C                    FREQUENCY
  310 INDEX=INDEX+1
      RMAG(INDEX)=1./CABS(R)
      FREQ(INDEX)=AIMAG(OMEGA3)
      GO TO 230
  320 IF(IQUAD3.EQ.IQUAD1) GO TO 330
      OMEGA2=OMEGA3
      GO TO 340
```

```
  330 OMEGA1=OMEGA3
      IQUAD1=IQUAD3
  340 CONTINUE
      WRITE(6,351)
  351 FORMAT(1HO,47H***WARNING***  FAILED TO CONVERGE TO ZERO PHASE)
      GO TO 230
C         OUTPUT LEAST UPPER AND GREATEST LOWER GAIN LIMITS
  400 IF(INDEX.EQ.0) GO TO 500
      GAMAX=1.E36
      GAMIN=-1.E36
      MAX=0
      MIN=0
      IMAX=0
      IMIN=0
C ========  SCAN STABILITY MARGINS THAT WERE LOCATED AND LEAST UPPER
C           BOUND AND GREATEST LOWER BOUND.
      DO 450 I=1,INDEX
      IF(RMAG(I).LT.1.) GO TO 430
      IF(RMAG(I).GT.GAMAX) GO TO 450
      GAMAX=RMAG(I)
      OMMAX=FREQ(I)
      MAX=1
      IMAX=I
      GO TO 450
  430 IF(RMAG(I).LT.GAMIN) GO TO 450
      GAMIN=RMAG(I)
      OMMIN=FREQ(I)
      MIN=1
      IMIN=I
  450 CONTINUE
      IF(MAX.NE.1) GO TO 405
      GAINL=GAMAX*GAINO
C =========  PRINT UPPER STABILITY MARGINS
      WRITE(6,540) GAINL,GAMAX,OMMAX
  540 FORMAT(1HO,10X,18HUPPER GAIN LIMIT =,G11.4, 5X,
     1   28H(UPPER GAIN LIMIT)/NOMINAL =,G11.4,5X,11HFREQUENCY =,G11.4,
     2   7H R.P.S.)
C ==========  LOAD SUMMARY ARRAY
      GMDSPY(INDEXF(K,4,KMAX))=GAMAX
      GMDSPY(INDEXF(K,5,KMAX))=OMMAX
  560 CONTINUE
      GO TO 407
  405 GMDSPY(INDEXF(K,4,KMAX))=1111.
      GMDSPY(INDEXF(K,5,KMAX))=1111.
      WRITE(6,406)
  406 FORMAT(1HO,10X,*NO UPPER LIMIT WAS LOCATED*)
  407 IF(MIN.NE.1) GO TO 470
      GAINL=GAMIN*GAINO
C ========  PRINT LOWER STABILITY MARGINS
      WRITE(6,410) GAINL,GAMIN,OMMIN
  410 FORMAT(1HO,10X,18HLOWER GAIN LIMIT =,G11.4, 5X,
     1  28H(LOWER GAIN LIMIT)/NOMINAL =,G11.4,5X,11HFREQUENCY =,G11.4,
     2      7H R.P.S.)
C ==========  LOAD SUMMARY ARRAY
      GMDSPY(INDEXF(K,2,KMAX))=GAMIN
      GMDSPY(INDEXF(K,3,KMAX))=OMMIN
      GO TO 475
  470 IF(IZERO.EQ.1) GO TO 475
```

```
      GMDSPY(INDEXF(K,2,KMAX))=1111.
      GMDSPY(INDEXF(K,3,KMAX))=1111.
      WRITE(6,473)
  473 FORMAT(1HO,10X,*NO LOWER LIMIT WAS LOCATED*)
  475 IF(MIN+MAX.EQ.INDEX) GO TO 600
C ===========  LIST OTHER NONCRITICAL STABILITY LIMITS
      WRITE(6,481)
  481 FORMAT(1HO,30X,29HOTHER NONCRITICAL GAIN LIMITS)
      DO 485 I=1,INDEX
      IF((I.EQ.IMAX).OR.(I.EQ.IMIN)) GO TO 485
      WRITE(6,484) RMAG(I),FREQ(I)
  484 FORMAT(1HO,20X,22H(GAIN LIMIT)/NOMINAL =,G11.4,5X,
     1   11HFREQUENCY =,G11.4)
  485 CONTINUE
      GO TO 600
  500 WRITE(6,501)
  501 FORMAT(1HO,10X,*NO LIMITS WERE LOCATED*)
      J=2
      IF(IZERO.EQ.1) J=4
      DO 502 I=J,5
  502 GMDSPY(INDEXF(K,I,KMAX))=1111.
  600 CONTINUE
C ==========  RESTORE SM PARAMETER TO NOMINAL VALUE
      CALL VARMOD(IACT(K),GAIMO)
C ========  TEST IF ALL SM PARAMETERS HAVE BEEN EVALUATED.
 2000 IF(K.LT.KMAX) GO TO 30
      RETURN
 2010 DO 2020 J=2,5
 2020 GMDSPY(INDEXF(K,J,KMAX))=0.
      GO TO 2000
      END
```

```
CGFBTCH
      OVERLAY(GFBTCH,3,0)
      PROGRAM GFBTCH
C  VERSION 3.                              REVISED: APRIL 30 1976
      COMMON /CP/P(1)/CX/X(1)/CXDOT/XDOT(1)/CXIC/XIC(1)
      COMMON/CORDER/NSIM,NDV,NOP/COVRLY/INST,LOKSS,LOKSIM
      COMMON/CPRON/DEPEN,INDEP1,INDEP2,DUM1(5)
      COMMON/CPROV/XMIN1,XMAX1,XMIN2,DELTA2,CURVES,DUM2(15)
      COMMON /CWORK/GDSPLY(50,2,10)/ERMESS/IFATAL,IERR
      COMMON/CNTRLS/ANTYPE,IPRIN,IMODE,ERROR(1)
      COMMON/CTIME/TIME
      COMMON /CPLOTS/ INDPLT,INDWR,IOPT(30),PLOTID( 5),PTITLE( 8),
     +                IPOPT(10)
      REAL   XOPT(1)
      EQUIVALENCE (XOPT(1),IOPT(1))
      REAL DEPEN,INDEP1,INDEP2
      TIME=0.
      DATA IBLNK /10H          /
      IF(INST.EQ.13)GO TO 4050
      CALL CODGEN(INDEP2,0,IND2),RETURNS(4000)
      ICUR=IFIX(CURVES)
 2000 CALL CODGEN(DEPEN,0,IDEPEN),RETURNS(4020)
      CALL CODGEN(INDEP1,0,IND1),RETURNS(4040)
      CALL VAROUT(IND2,XDUM2)
      IF(ICUR.LT.1) ICUR=1
      IF(ICUR.GT.10) ICUR=10
      YMIN1=1.E36
      YMAX1=-1.E36
      CALL VAROUT(IND1,XDUM)
      NPTS=50
      DELTA=(XMAX1-XMIN1)/49.
      DO 2135 I=1,NSIM
      XDOT(I)=0.
 2135 X(1)=XIC(I)
      IOPT(3) = IBLNK
      IOPT(4) = IBLNK
      CALL DTTIM (IOPT(3))
      ISET=1
C ------------  TURN ON ERROR MESSAGES IN MODEL
      IERR=1
      CALL EQMO(TIME  ,TIME  ,ISET)
      XDUM1=XMIN2-DELTA2
      IF(ICUR.GT.1) GO TO 2130
      WRITE(6,2131)DEPEN,INDEP1,PTITLE,IOPT(3),IOPT(4)
 2131 FORMAT(40X,46H /*/*/*/   GENERAL FUNCTION ANALYSIS   /*/*/*/
     1//55X,A8,4HVS  ,A8//26X,8A10//54X,2A12/)
      GO TO 2137
 2130 WRITE(6,2134) DEPEN,INDEP1,INDEP2,PTITLE,IOPT(3),IOPT(4)
 2134 FORMAT(40X,46H /*/*/*/   GENERAL FUNCTION ANALYSIS   /*/*/*/  ,
     1//50X,A8,4HVS  ,A8,2H+ ,A8 //26X,8A10//54X,2A12/)
 2137 DO 2139 J=1,ICUR
      XDUM1=XDUM1+DELTA2
      IF(ICUR.GT.1)WRITE(6,2132)J,INDEP2,XDUM1
 2132 FORMAT(*0 CURVE NO.*,I3,4X,A8,3H = ,G12.5)
      IF(ICUR.GT.1) CALL SETIN(IND2,XDUM1)
      XOPT(J+15) = XDUM1
      DO 2133 I=1,NPTS
      GDSPLY(I,2,J)=XMIN1+(I-1)*DELTA
```

```
      CALL SETIN(IND1,GDSPLY(I,2,J))
      CALL VAROUT(IDEPEN,GDSPLY(I,1,J))
      IF(GDSPLY(I,1,J).GT.YMAXI) YMAXI=GDSPLY(I,1,J)
      IF(GDSPLY(I,1,J).LT.YMINI) YMINI=GDSPLY(I,1,J)
 2133 CONTINUE
      WRITE(6,2138)(INDEP1,(GDSPLY(I+(K-1)*10,
     1 2,J),I=1,10),DEPEN,(GDSPLY(I+(K-1)*10,1,J),I=1,10),
     2 K=1,5)
 2138 FORMAT(1H ,A8,1H:,10G12.5)
 2139 CONTINUE
C
C     SET PLOT PARAMETERS
C
      IF ( INDPLT .EQ. 0 ) GO TO 3500
      IOPT(1) = 1
      IOPT(2) = IOPT(2) + 1
      XOPT(5) = DEPEN
      XOPT(6) = INDEP1
      XOPT(7) = INDEP2
      DO 3100 I=8,13
 3100 IOPT(I) = 0
      IOPT(14) = NPTS
      IOPT(15) = ICUR
      WRITE (30) IOPT,PLOTID,PTITLE
      WRITE (30) (((GDSPLY(I,J,K),I=1,NPTS),J=1,2),K=1,ICUR)
      INDWR = 1
 3500 CONTINUE
      CALL VARMOD(IND1,XDUM)
      IF(ICUR.LE.1) GO TO 2150
      CALL VARMOD(IND2,XDUM2)
2150  WRITE(6,2151)
2151  FORMAT(/////)
      GO TO 6000
 4000 WRITE(6,4001) INDEP2
 4001 FORMAT(//10X,31H*** WARNING ***  CAN*T IDENTIFY,1X,A10,1X,
     1 25HAS A VALID SCAN PARAMETER//)
      CALL CODGEN(DEPEN,0,IDEPEN),RETURNS(4020)
 4010 CALL CODGEN(INDEP1,0,IND1),RETURNS(4040)
      WRITE(6,2151)
      GO TO 6000
 4020 WRITE(6,4001) DEPEN
      GO TO 4010
 4040 WRITE(6,4001) INDEP1
      WRITE(6,2151)
      GO TO 6000
4050  ICUR=1
      GO TO 2000
6000  CONTINUE
C ————————  TURN OFF ERROR MESSAGES IN MODEL
      IERR=0
      END
```

```
CINIT
      OVERLAY(INIT,1,0)
      PROGRAM INIT
C  VERSION 1.2                              REVISED: MAY 15 1975
C  PURPOSE:  TO INITIALIZE INTEGRATOR CONTROL,PARAMETER NAME,STATE
C            NAME, RATE NAME, VARIABLE NAME ARRAYS TO DEFAULT VALUES
C  DESIGNED BY: J.D. BURROUGHS                        FEB 1974
      COMMON /CORDER/NOX,NOV,NOP/CINT/INT(1)
      COMMON/CNAMEX/NAMEX(1)/CNAMER/NAMER(1)/CNAMEV/NAMEV(1)/CNAMEP/
     1 NAMEP(1)/CXIC/XIC(1)
      COMMON/CNTRLS/ANTYPE,IPRINT,MODE,ERROR(1)
      COMMON/CWORKN/NN,N(7)
      REAL    NAMEX,NAMER,NAMEV,NAMEP
C  INITIALIZE INT ARRAY
      DO 10 I=1,NOX
      ERROR(I)=.1
      XIC(I)=0.
   10 INT(I)=1
C  LOAD STATE NAME ARRAY WITH S001,S002,....
C     CALL CODLOD(NAMEX,NOX,1HS)
C  LOAD RATE NAME ARRAY WITH R001,R002,....
      CALL CODLOD(NAMER,NOX,1HR)
C  LOAD PARAMETER NAME ARRAY WITH P001,P002,...
C     CALL CODLOD(NAMEP,NOP,1HP)
C  LOAD VARABLE NAME ARRAY WITH V001,V002,...
C     CALL CODLOD(NAMEV,NOV,1HV)
C  CALCULATE INDICES FOR WORK  STORAGE
      NN=NOX*NOX+1
      N(1)=NN+NOX*NOX
      IF(N(1).LT.168)N(1)=168
      DO 100 I=2,7
  100 N(I)=N(I-1)+NOX
      CALL PLINIT
      END
```

```
CINPUTS
      SUBROUTINE INPUTS(A,N,M,NMAX)
C  VERSION 1.                          REVISED: MAY 22 1975
C  PURPOSE:  ALLOW FREE FIELD INPUT OF ARRAY DATA
C  CALL SEQUENCE:  A      - ARRAY TO RECEIVE DATA
C                  N      - NUMBER OF ROWS IN ARRAY
C                  M      - NUMBER OF COLUMNS IN ARRAY
C                  NMAX   - ROW DIMENSION OF ARRAY A
C  METHOD:  THE FOLLOWING COMMANDS ARE RECOGNIZED
C         Z = ZERO ALL ELEMENTS OF ARRAY
C         I = SET ALL ELEMENTS OF ARRAY TO 1.E36 (INFINITY)
C         C = INPUT DATA TO BE GIVEN BY COLUMN
C         R = INPUT DATA TO BE GIVEN BY ROW
C         D = INPUT DATA TO BE GIVEN BY DIAGONAL
C     FOLLOWING THE COL, ROW, DIAG, COMMANDS THE ROW AND COLUMN LOCATION
C     AT WHICH DATA LOADING IS TO START MUST BE GIVEN.  THESE VALUES
C     ARE FOLLOWED BY ELEMENT VALUES.  EACH COMMAND, ROW NO., COL. NO.,
C     OR ELEMENT VALUE MUST BE SEPERATED BY ONE OF THE STANDARD DELIMITE
C     STANDARD DELIMITERS ARE: THREE OR MORE SPACES; COMMA; EQUAL SIGN;
C     LEFT OR RIGHT PARENTHESIS.
C  DESIGNED BY: J.D.BURROUGHS                    MAY 1975
      COMMON/CCOMM/ICOM(8),IPHRS,INDEX
      DIMENSION ICOML(5),A(1)
      DATA ICOML/50H2         I         C         R         D         /
C ---->     SET DEFAULT MODE TO COLUMN INPUT
      MODE=3
C =======  MODE = MODE OF INPUT INDICATOR. 1 = ZERO ARRAY
C         2 = SET ARRAY TO 1.E36, 3 = COLUMN INPUT, 4 = ROW INPUT,
C         5 = DIAGONAL INPUT.
      I=1
      J=1
      ISTAT=2
C ========  ISTAT = INPUT STATUS INDICATOR.  0 = ROW NO. NEEDED; 1 = COL
C                                           2 = READY FOR DATA VALUES
100   INDEXS=INDEX
C ---->     LOCATE NEXT PHRASE
      CALL NXTPH(ICOM,INDEX,IPHRS)
      IF(IPHRS.NE.1OH          )GO TO 200
C ---->     READ NEXT CARD
      READ(5,121)ICOM
      IF(EOF(5)) 520,140
121   FORMAT(8A10)
140   WRITE(6,141)ICOM
141   FORMAT(/20H COMMAND CARD ----->,5X,8A10)
      INDEX=1
      GO TO 100
C ---->     TEST FOR NUMERIC PHRASE
200   CALL NUMERC(IPHRS),RETURNS(300)
C ---->     NUMERIC PHRASE DETECTED
      CALL BCDREL(VALUE,IPHRS)
      IF(ISTAT-1)210,220,240
210   I=VALUE
      ISTAT=1
      GO TO 100
220   J=VALUE
      ISTAT=2
      GO TO 100
C ---->     TESTS TO LIMIT INPUT TO GIVEN ROW AND COLUMN DIMENSIONS
```

```
240   IF(I.GT.N.OR.J.GT.M)GO TO 100
      K=I+NMAX*(J-1)
      A(K)=VALUE
C ---->       INCREASE INDICES DEPENDING ON INPUT MODE
      IF(MODE-4)280,260,270
260   J=J+1
      GO TO 100
270   J=J+1
280   I=I+1
      GO TO 100
C ---->      ALPHA PHRASE DETECTED
300   CALL LCMPH(IPHRS,ICOML,5,1,MODE)
      IF(MODE.EQ.0)GO TO 500
C ---->       RESTORE INDEX TO PREVIOUS PHRASE SINCE ALPHA PHRASE IS NOT R
      IF(MODE-2)340,380,310
310   ISTAT=0
      GO TO 100
C ---->       ZERO ARRAY MODE
340   NM=NMAX*M
      DO 360 I=1,NM
360   A(I)=0.
      GO TO 100
C ---->       SET ARRAY TO 1.E36 (INFINITY)
380   NM=NMAX*M
      DO 400 I=1,NM
400   A(I)=1.E36
      GO TO 100
500   INDEX=INDEXS
520   RETURN
      END
```

```
CINTERP
      OVERLAY(INTERP,2,0)
      PROGRAM INTERP
C  VERSION 4.                          REVISED: JULY 6 1977
C       PURPOSE:
C      READS,PRINTS AND INTERPRETS INSTRUCTIONS FROM DATA CARDS
C       CALL SEQUENCE:
C          IREAD - READ UNIT NUMBER
C          INST  - INSTRUCTION NUMBER
C  DESIGNED BY: J.D. BURROUGHS                    FEB 1974
      DIMENSION AINT(1)
      COMMON /CNTRLS/INSTO ,IPRINT,IMODE,ERROR(1)
      COMMON /COVRLY/INST,LOKSS,LOKSIM,CPUSEC/CIO/IREAD,IWRITE,IDIAG
      COMMON/CXIC/XIC(1)/CWORK/WORK(1)/CP/P(1)/CINT/INT(1)/CX/X(1)
      COMMON /CXIC1/XIC1(1)/CXIC2/XIC2(1)/CXIC3/XIC3(1)
      COMMON/CNAMEX/NAMEX(1)/CNAMER/NAMER(1)/CNAMEV/NAMEV(1)/CNAMEP/
     1 NAMEP(1)
      COMMON/CUNITX/NUNITX(1)/CUNITR/NUNITR(1)/CUNITV/NUNITV(1)/CUNITP/
     1 NUNITP(1)
      COMMON/CSCALE/SCALE(5,4,6),NVAR(5,2,6),NPLTS(6)
      COMMON /CSMPAR/SMPAR(10),ICIND(2)
      COMMON/CORDER/NOX,NOV,NOP/CTIME/TIME
      COMMON/CPRINT/PRTNAM(10),LPRT(10)
      COMMON /CPRON/PRONAM(8)/CPROV/PVALUE(27)
      COMMON /CPLOTS/ INDPLT,INDWR,IOPT(30),PLOTID( 5),PTITLE( 8),
     +                IPOPT(10)
      COMMON/CCOMM/ICOM(8),IPHRS,INDEX
      COMMON/COLDIM/NX,NU,NS,NC,NRS,NRC,IXOC,IUOC,IOCAN,IPOINT(25)
      COMMON/CTABNA/TABNAM(1)/CMAXDI/NOTAB,MAXDIM(1)/CLOCTA/LOCTAB(1)
      COMMON/CTABLE/TABLES(1)
      REAL    IPHRS,ICOML(59),NAMEX,NAMER,NAMEV,NAMEP,NUNITX,NUNITR,
     1 NUNITV,NUNITP
      REAL    NVAR,IBLNK,IPROGN(8),PRONAM,IPROGV(27),SMPAR,IC
      EQUIVALENCE (AINT,INT)
      DATA ICLMAX/59/,NONE/1OHNONE      /
      DATA IPNMAX/8/,IPVMAX/27/
C ================  PROGRAM COMMANDS  ====================
      DATA ICOML /59OHDEFINE STADEFINE RATDEFINE PARDEFINE VARINITIAL CO
     1PARAMETER DISPLAY1  DISPLAY2  DISPLAY3  DISPLAY4  DISPLAY5  DISPLA
     2Y6  SCAN1      SCAN2      XIC-X      XIC-XIC1  XIC-XIC2  XIC-XIC3  XI
     3C1-XIC  XIC2-XIC  XIC3-XIC  ALL STATESNO STATES INT CONTROERROR CO
     4NTSIMULATE  LINEAR ANAEIGEN SENSSTABILITY TRANSFER FSTEADY STAROOT
     5 LOCUSPUNCH X    SM PARAMETPLOT TABLEPRINT VARITITLE      PLOT ID
     6PLOT ON    PLOT OFF  SC4020     CALCOMP    RL MANUAL RL AUTO SCSI MAN
     7UAL SI AUTO SCSS MANUAL SS AUTO SCTF MANUAL TF AUTO SCBODE       NI
     8CHOLS    NYQUIST    PRINTER PLDESIGN O.CO.C. DATA SAVE O.C. PLOT ALL
     9 TTABLE      /
      DATA IBLNK/1OH           /,IC/1OHIC        /
C ================  PROGRAM NAMES  ====================
      DATA IPROGN/8OHDEPEN      INDEP1     INDEP2     EIGEN PARATF INPUT  T
     1F OUTPUT SS PARAMETRL PARAMET/
C ================  PROGRAM VALUES  ====================
      DATA IPROGV/27OHSTART1     STOP1      START2     DELTA2     CURVES2
     1PRINT CONTPRATE      OUTRATE    INT MODE  TINC       TMAX       FREQ M
     2AX  FREQ MIN  SS START  SS STOP   SS POINTS SS ITERATIRL START  RL
     3 STOP   RL POINTS REAL MIN  REAL MAX  IMAG MIN  IMAG MAX  O.C. MOD
     4ELO.C. ORDERINITIAL TI/
C ------  TEST FOR CPU SECOND MEASURE
```

```
      IF(CPUSEC.EQ.0.)GO TO 80
      CALL SECOND(CPSEC)
      CPDEL=CPSEC-CPUSEC
      WRITE(6,71)CPDEL
71    FORMAT(/10X,G13.6,* CPU SECONDS WERE REQUIRED FOR THE PREVIOUS ANA
     1LYSIS*/)
80    NAMPRT=INST
      IMODE=PVALUE(9)
90    INSTO=0
      IF(INDEX.GT.0.AND.INDEX.LT.81)GO TO 120
C============== READ AND WRITE ONE CARD  =========
 100  READ(IREAD,101) ICOM
      IF(EOF(5)) 5000,111
101   FORMAT(8A10)
 111  WRITE(6,105) ICOM
105   FORMAT(/20H COMMAND CARD ------>,5X,8A10)
C------>SET CHARACTER SCAN INDEX
      INDEX=1
C------>LOCATE NEXT PHRASE
120   CALL NXTPH(ICOM,INDEX,IPHRS)
C------>READ NEXT CARD IF BLANK PHRASE
140   IF(IPHRS.EQ.IBLNK) GO TO 100
C------>SEARCH COMMAND LIST
      CALL LCMPH(IPHRS,ICOML,ICLMAX,1,INST)
C------>COMMAND IDENTIFIED
      IF(INST.LE.0) GO TO 160
C========== BRANCH TO NEW COMMAND  ==========
      GO TO (200,200,200,200,200,200,210,220,230,232,
     1       234,236,500,500,240,250,260,270,280,290,
     2       300,310,320,200,200,500,500,500,500,500,
     3       500,500,920,330,200,360,550,560,570,580,
     4       590,600,610,620,630,640,650,660,670,680,
     5       690,700,710,720,820,800,900,960,980)                ,INST
C===========  SEARCH PROGRAM NAME LIST  ==========
  160 CALL LCMPH(IPHRS,IPROGN,IPNMAX,1,INST)
C------>PHASE NOT PROGRAM NAME
      IF(INST.LE.0) GO TO 170
C------>GET NEXT PHRASE
      CALL NXTPH(ICOM,INDEX,IPHRS)
C------>LOAD PROGRAM NAME
      PRONAM(INST)=IPHRS
      IF(INST.NE.7.AND.INST.NE.8) GO TO 165
      IF(IPHRS.EQ.NONE)PRONAM(INST)=IBLNK
C------>GET NEXT PHRASE
      CALL NXTPH(ICOM,INDEX,IPHRS)
      ICIND(INST-6)=0
      IF(IPHRS.NE.IC) GO TO 168
C------>SET INDICATER .EQ. 1
      ICIND(INST-6)=1
  165 INSTO=0
      GO TO 120
  168 INSTO=0
      GO TO 140
C========  SEARCH PROGRAM VALUE LIST  =========
  170 CALL LCMPH(IPHRS,IPROGV,IPVMAX,1,INST)
C------>PHRASE NOT PROGRAM VALUE
      IF(INST.LE.0) GO TO 178
C------>GET NEXT PHRASE
```

```
      CALL NXTPH(ICOM,INDEX,IPHRS)
C------>TEST 1ST CHARACTER FOR NUMERIC
      CALL NUMERC(IPHRS),RETURNS(176)
C------>CONVERT A TO G FORMAT
      CALL BCDREL(PVALUE(INST),IPHRS)
      GO TO 165
  176 WRITE(6,177) IPROGV(INST),IPHRS
  177 FORMAT(//10X,15H*** WARNING ***,3X,A10,22HCAN*T BE SET EQUAL TO:,
     1 A10,23H  VALUE MUST BE NUMERIC //)
      GO TO 168
C------>CHECK FOR OUTSTANDING COMMAND
  178 IF(INSTO.LE.0) GO TO 180
C=========  BRANCH TO OUTSTANDING COMMAND  ==========
      GO TO (410,420,430,440,450,460,480,480,480,480,
     1        480,480,500,500,240,250,260,270,280,290,
     2        300,310,320,520,530,500,500,500,500,500,
     3        500,500,500,540,940,545,550,560,500,500,
     4        500,500,500,500,500,500,500,500,500,500,
     5        500,500,500,500,500,600,800,960,980)          ,INSTO
180   WRITE(6,181)IPHRS
181   FORMAT(//15X,34H*** WARNING ***  CAN*T INTERPRET  ,A10//)
      GO TO 120
C------>SET INSTO TO INDICATE A NEW OUTSTANDING TASK
200   INSTO=INST
      MODE=-1
      GO TO 120
210   IDSPLY=1
215   NPLTS(IDSPLY)=0
      GO TO 200
220   IDSPLY=2
      GO TO 215
230   IDSPLY=3
      GO TO 215
  232 IDSPLY = 4
      GO TO 215
  234 IDSPLY = 5
      GO TO 215
  236 IDSPLY = 6
      GO TO 215
C------>TRANSFER X TO XIC
  240 CALL XFR(X,XIC,NOX)
      LOKS1M=LOKSS
  245 WRITE(6,2630)(I,NAMEX(I),XIC(I),I=1,NOX)
 2630 FORMAT(1H1,40X,7H/*/*/*/,3X,*INITIAL CONDITIONS/OPERATING POINT*,
     1 3X,7H/*/*/*/,//5(I4,1H ,A8,3H = ,G10.4))
      WRITE(6,247)
247   FORMAT(/////)
      GO TO 165
C ==========  TRANSFER XIC1 TO XIC  ==========
  250 CALL XFR(XIC1,XIC,NOX)
      GO TO 245
C ==========  TRANSFER XIC2 TO XIC  ==========
  260 CALL XFR(XIC2,XIC,NOX)
      GO TO 245
C ==========  TRANSFER XIC3 TO XIC  ==========
  270 CALL XFR(XIC3,XIC,NOX)
      GO TO 245
C ==========  TRANSFER XIC TO XIC1  ==========
```

```
  280 CALL XFR(XIC,XIC1,NOX)
      GO TO 165
C ===========  TRANSFER XIC TO XIC2  ==========
  290 CALL XFR(XIC,XIC2,NOX)
      GO TO 165
C ===========  TRANSFER XIC TO XIC3  ==========
  300 CALL XFR(XIC,XIC3,NOX)
      GO TO 165
C ===========  ALL STATES  ===========
  310 DO 315 I=1,NOX
  315 INT(I)=1
      GO TO 165
C ===========  NO STATES  ===========
  320 DO 325 I=1,NOX
  325 INT(I)=0
      GO TO 165
C------>LOAD SMPAR WITH BLANKS
  330 DO 335 I=1,10
  335 SMPAR(I)=IBLNK
338   INSTO=INST
      ITNO=1
  340 MODE=0
      GO TO 120
C-----LOAD PRTNAM WITH BLANKS
360   DO 365 I=1,10
      LPRT(I)=-1
365   PRTNAM(I)=IBLNK
      GO TO 338
C------>DEFINE STATES TASK
410   CALL NAMES(IPHRS,NAMEX,NUNITX,NOX,ITNO,MODE)
      GO TO 120
C------>DEFINE RATES TASK
420   CALL NAMES(IPHRS,NAMER,NUNITR,NOX,ITNO,MODE)
      GO TO 120
C------>DEFINE PARAMETERS TASK
430   CALL NAMES(IPHRS,NAMEP,NUNITP,NOP,ITNO,MODE)
      GO TO 120
C------>DEFINE VARIABLES TASK
440   CALL NAMES(IPHRS,NAMEV,NUNITV,NOV,ITNO,MODE)
      GO TO 120
C------>INITIAL CONDITIONS TASK
450   CALL VALUES(IPHRS,NAMEX,NOX,XIC,ITNO,MODE)
      GO TO 120
C------>PARAMETER INPUT TASK
460   CALL VALUES(IPHRS,NAMEP,NOP,P,ITNO,MODE)
      GO TO 120
C------>DISPLAY TASK
480   CALL DISPLA(IDSPLY,IPHRS,MODE,ICOL)
      GO TO 120
C+--->RETURN TO MAIN PROGRAM WITH INST SET TO INDICATED TASK
 500  INSTO=0
      IF(NAMPRT.EQ.1)GO TO 5005
      GO TO 6000
C------>LOAD INTEGRATOR CONTROLS
  520 CALL VALUES(IPHRS,NAMEX,NOX,AINT,ITNO,MODE)
C------>CONVERT REAL TO INTEGER
      IF(MODE.EQ.0) INT(ITNO)=AINT(ITNO)
      GO TO 120
```

```
C------>LOAD ERROR CONTROLS
  530 CALL VALUES(IPHRS,NAMEX,NOX,ERROR,ITNO,MODE)
      GO TO 120
C------>LOAD STABILITY MARGIN PARAMETER NAME
  540 CALL NAMES(IPHRS,SMPAR,NUNIT,10,ITNO,MODE)
542   ITNO=ITNO+1
      GO TO 340
C------ LOAD PRINT VARIABLE NAMES
  545 CALL NAMES(IPHRS,PRTNAM,NUNIT,10,ITNO,MODE)
C------  DETERMINE I.D. CODES FOR PRINT QUANTITIES
      IF(MODE.NE.1)GO TO 542
      CALL CODGEN(PRTNAM(ITNO),0,LPRT(ITNO)),RETURNS(546)
      GO TO 542
546   WRITE(6,547)PRTNAM(ITNO)
547   FORMAT(//20X,31H*** WARNING ***  CAN T IDENTIFY,3X,A10
     1,*AS A VALID PRINT VARIABLE*//)
      GO TO 542
C
C     SET PLOTTING OPTIONS
C
C ==========  TITLE  ==============
550   CALL TITLE (ICOM,INDEX,PTITLE,80)
      GO TO 562
C ==========  PLOT ID  ==============
560   CALL TITLE (ICOM,INDEX,PLOTID,48)
562   INDEX=0
      GOTO 90
C ==========  PLOT ON  ===========
  570 INDPLT = 1
      CALL ONSW(1)
      GO TO 165
C ==========  PLOT OFF  ===========
  580 INDPLT = 0
      GO TO 165
C ==========  SC4020  =========
  590 IOPT(29) = 0
      GO TO 165
C ==========  CALCOMP  ============
  600 IOPT(29) = 1
      GO TO 165
C ==========  RL MANUAL SCALES  ==========
  610 IPOPT(1) = 1
      GO TO 165
C =========  RL AUTO SCALES  ========
  620 IPOPT(1) = 0
      GO TO 165
C ============  SI MANUAL SCALES  ============
  630 IPOPT(2) = 1
      GO TO 165
C ==========  SI AUTO SCALES  ===========
  640 IPOPT(2) = 0
      GO TO 165
C ==========  SS MANUAL SCALES  =========
  650 IPOPT(3) = 1
      GO TO 165
C =========  SS AUTO SCALES  =========
  660 IPOPT(3) = 0
      GO TO 165
```

```
C ==========  TF MANUAL SCALES  =========
  670 IPOPT(4) = 1
      GO TO 165
C ========  TF AUTO SCALES  =========
  680 IPOPT(4) = 0
      GO TO 165
C ==========  BODE  =========
  690 IPOPT(5) = 1
      IPOPT(6)=0
      IPOPT(7)=0
      GO TO 165
C ==========  NICHOLS  ==========
  700 IPOPT(7) = 1
      IPOPT(6)=0
      IPOPT(5)=0
      GO TO 165
C =========  NYQUIST  =======
  710 IPOPT(6) = 1
      IPOPT(7)=0
      IPOPT(5)=0
      GO TO 165
C ==========  PRINTER PLOTS  ==========
  720 IOPT(30) = 1
      CALL ONSW(2)
      INDPLT=1
      GO TO 165
C ------      READ O.C. DATA TASK
800   CALL OCDATA
      GO TO 165
C ============  DESIGN O.C. TASK  =============
C ---       TEST THAT MODEL IS DIMENSIONED FOR O.C. DESIGN
820   IF(IOCAN.EQ.2)GO TO 500
      WRITE(6,825)
825   FORMAT(//15X,15H*** WARNING ***,3X,*WORK SPACE WAS NOT PROVIDED IN
     1 MODEL FOR OPTIMAL CONTROLLER DESIGN*//)
      GO TO 165
C =============  SAVE O.C. TASK ============
900   CALL OCSAVE
      GO TO 165
C =============  PUNCH X TASK  ===============
920   WRITE(3,921)
921   FORMAT(*INITIAL CONDITIONS*)
      WRITE(3,922)(NAMEX(I),X(I),I=1,NOX)
922   FORMAT(4(A7,*=*,G10.4,*,*))
      GO TO 165
C =============  PLOT TABLES TASK  ===========
940   CALL LCMPH(IPHRS,TABNAM,NOTAB,1,NTAB)
      IF(NTAB.LE.0)GO TO 950
C-------     CALL TABLE PLOTTING ROUTINE
945   CALL PLOTAB(NTAB)
      CALL ONSW(1)
      GO TO 120
950   WRITE(6,951)IPHRS
951   FORMAT(//15X,15H*** WARNING ***,3X,A10,* IS NOT VALID TABLE NAME*
     1//)
      GO TO 120
C ==============   PLOT ALL TABLES  TASK   ============
960   NTAB=-1
```

```
      GO TO 945
C ==============   TABLE   TASK   ================
980   BACKSPACE IREAD
      CALL TABIN(TABLES,TABNAM,MAXDIM,LOCTAB,NOTAB)
      INDEX=0
      GO TO 90
C------>END OF FILE ENCOUNTERED
5000  INST=-1
5005  WRITE(6,5010)(I,NAMEX(I),I=1,NOX)
 5010 FORMAT(//1H1,50X,11HSTATE NAMES//10(I4,1X,A8))
      WRITE(6,5020)(I,NAMER(I),I=1,NOX)
 5020 FORMAT(//50X,10HRATE NAMES//10(I4,1X,A8))
      WRITE(6,5030)(I,NAMEV(I),I=1,NOV)
 5030 FORMAT(//50X,14HVARIABLE NAMES//10(I4,1X,A8))
      WRITE(6,5050)(I,NAMEP(I),P(I),I=1,NOP)
5050  FORMAT(//49X,*PARAMETER VALUES*//5(I4,1X,A8,
     12H= ,G11.5))
C ============  SCAN FOR UNINITIALIZED PARAMETERS
      J=0
      DO 5100 I=1,NOP
      IF(P(I).NE..99999)GO TO 5100
      J=J+1
      WORK(J)=NAMEP(I)
5100  CONTINUE
      IF(J.GT.0)WRITE(6,5101)(WORK(I),I=1,J)
5101  FORMAT(////15X,15H*** WARNING ***,15X,*UNINITIALIZED PARAMETERS*
     1 //10(3X,A8,2X))
6000  CONTINUE
      TIME=PVALUE(27)
      WRITE(6,6001)
6001  FORMAT(1H1)
C ------          GET CURRENT CPU TIME
      CALL SECOND(CPUSEC)
      END
```

```
CLABTC
      OVERLAY(LABTCH,5,0)
      PROGRAM LABTC
C  PURPOSE:  PROVIDE OVERLAY INTERFACE TO PASS WORK STORAGE
C              ARRAYS TO LINEAR ANALYSIS ROUTINES LABTCH AND ESBTCH.
C  DESIGNED BY: J.D. BURROUGHS                    FEB 1974
C  VERSION 1.                      REVISED: JUNE 9 1975
      COMMON/CORDER/NSIM,NOV,NOP
      COMMON/CPRON/DUM1(3),ESPAR,DUM2(4)
      COMMON/COVRLY/INST,LOKSS,LOKSIM
      COMMON/CWORK/A(1)
      COMMON /CWORKN/NN,N1,N2,N3,N4,N5,N6,N7
      REAL ESPAR
      IF(INST.EQ.28)GO TO 100
      CALL LABTCH(NSIM,ESPAR,A,A(NN),A(N1),A(N2),A(N3),A(N4),A(N5),A(N6)
     1,A(N7),A(N2),A(N3),A(N4),A(N5),A(N6),A(N7),A(N1),A(N1))
      GO TO 6000
100   CALL ESBTCH(NSIM,ESPAR,A,A(NN),A(N1),A(N2),A(N3),A(N4),A(N5),A(N6)
     1,A(N7),A(N2),A(N3),A(N4),A(N5),A(N6),A(N7),A(N1),A(N1))
6000  CONTINUE
      END
```

```
CLABTCH
      SUBROUTINE LABTCH(NSIM,ESPAR,A,RATIO,DWORK,IA,IB,IC,ID,IAP,IBP,
     1 EVR,EVI,WN,DAMPR,EVRP,EVIP,IRATIO,XDOTO)
C  VERSION  3.1                        REVISED: OCT 11 1976
C  PURPOSE:  PERFORM LINEAR ANALYSIS AND EIGENVALUE SENSITIVITY
C            ANALYSES.
C  CALL SEQUENCE:  NSIM  - MODEL ORDER
C                  ESPAR - EIGEN SENSITIVITY PARAMETER
C               NAME              DESCRIPTION          LOCATION
C                 A      - NSIM X NSIM  WORK ARRAY     /CWORK/A(1)
C                 RATIO  - NSIM X NSIM  WORK ARRAY     /CWORK/A(NN)
C                 DWORK  - NSIM X 1     WORK ARRAY    /CWORK/A(N1)
C                 IA     - NSIM X 1     WORK ARRAY     /CWORK/A(N2)
C                 IB     - NSIM X 1      WORK ARRAY    /CWORK/A(N3)
C                 IC     - NSIM X 1     WORK ARRAY     /CWORK/A(N4)
C                 ID     - NSIM X 1     WORK ARRAY     /CWORK/A(N5)
C                 IAP    - NSIM X 1     WORK ARRAY     /CWORK/A(N6)
C
C                 IBP    - NSIM X 1     WORK ARRAY     /CWORK/A(N7)
C                 EVR    - NSIM X 1     WORK ARRAY     /CWORK/A(N2)
C                 EVI    - NSIM X 1     WORK ARRAY     /CWORK/A(N3)
C                 WN     - NSIM X 1     WORK ARRAY     /CWORK/A(N4)
C                 DAMPR  - NSIM X 1     WORK ARRAY     /CWORK/A(N5)
C                 EVRP   - NSIM X 1     WORK ARRAY    /CWORK/A(N6)
C                 EVRI   - NSIM X 1     WORK ARRAY     /CWORK/A(N7)
C                 IRATIO- NSIM X 1      WORK ARRAY     /CWORK/A(N1)
C                 XDOTO  - NSIM X 1     WORK ARRAY       /CWORK/A(N1)
C  DESIGNED BY: J.D. BURROUGHS                FEB 1974
      COMMON /CXIC/XIC(1 )/CINT/INT(1 )
      COMMON /CNTRLS/ANTYPE,IPRINT,MODE,ERROR(1)
      COMMON /CNAMEX/NAMEX(1)/CNAMER/NAMER(1)
      REAL NAMEX,NAMER,ESPAR
      DIMENSION DWORK(1),IA(1),IB(1),IC(1),ID(1),IRATIO(1)
      REAL EVR(1),EVI(1),XDOTO(1),A(1),RATIO(1)
      DIMENSION WN(1),DAMPR(1)
      DIMENSION EVRP(1),EVIP(1),IAP(1),IBP(1)
      DATA IPOM/10H+-+-+-+-+-/
      DATA IASTRX/5(2H**)/,IBLNK/10H          /
      INDEX2(I1,I2,M1)=I1+(I2-1)*M1
C                     *** LINEAR ANALYSIS CALCULATIONS ***
 2800 WRITE(6,2802)
 2802 FORMAT(1H1/40X,7H/*/*/*/,3X,*LINEAR ANALYSIS*,3X,7H/*/*/*//)
      WRITE(6,2809)(I,NAMEX(I),XIC(I),ERROR(I),INT(I),I=1,NSIM)
 2809 FORMAT(/
     17X,5HSTATE,6X,*OPERATING   PERTURBATION   INTEGRATOR*
     2/8X,*NAME*,8X,*POINT*,9X,*SIZE*,8X,*CONTROL*/
     2 (I5,1X,A10,G14.5,G12.3,I9))
C ========  CALCULATE SYSTEM STABILITY MATRIX
      CALL STABMX(NSIM,XDOTO,ICOUNT,RATIO,A,NLIN,1)
      WRITE(6,2813)(I,NAMEX(I),XDOTO(I),I=1,NSIM)
 2813 FORMAT(///50X,*RATES AT OPERATING POINT*/5(I4,1H ,A8,3H = ,G11.5))
C ========  TEST IF ANY NONLINEAR ELEMENTS WERE DETECTED
      IF(ICOUNT.EQ.0) GO TO 2850
      WRITE(6,2849) ICOUNT
 2849 FORMAT(///5X,I4,2X,101HELEMENTS OF /RATIO/ DIFFER FROM 1 BY 10/. T
     1HESE ELEMENTS ARE PRECEDED BY AN * IN THE STABILITY MATRIX  )
      DO 2842 I=1,NLIN
      DO 2842 J=1,NLIN
```

```
      J2=INDEX2(I,J,NLIN)
      IF(ABS(RATIO(J2)-1.).GT..1) WRITE(6,2841) I,J,RATIO(J2)
 2841 FORMAT(20X,*RATIO(*,I2,*,*,I2,*) =*,G12.6)
 2842 CONTINUE
C ==========  PRINT STABILITY MATRIX
2850  K=0
      DO 2855 I=1,NSIM
      IF(INT(I).EQ.0)GO TO 2855
      K=K+1
      WN(K)=NAMEX(I)
2855  CONTINUE
      WRITE(6,2851)(WN(K),K=1,NLIN)
 2851 FORMAT(/// 47X,*STABILITY MATRIX*/(T10,10(2X,A7,3X)))
      DO 2852 I=1,NLIN
      DO 2853 K=1,NLIN
      IRATIO(K)=IBLNK
      IF(ABS(RATIO(INDEX2(I,K,NLIN))-1.).GT..1) IRATIO(K)=IASTRX
 2853 CONTINUE
 2852 WRITE(6,2843)WN(I),(IRATIO(J),A(I+(J-1)*NLIN),J=1,NLIN)
 2843 FORMAT(1X,A7,(T10,10(1X,A1,G10.4)))
C ========  CALCULATE EIGENVALUES AND NATURAL FREQUENCIES
2863  CALL EGVL3(A,RATIO,EVR,EVI,IA,IB,IC,ID,DWORK,1.E-14,NLIN,NLIN)
      CALL NATFRQ(EVR,EVI,WN,DAMPR,NLIN,NPOLES)
      WRITE(6,2867) NLIN
 2867 FORMAT(/// 28X,I3,2X,*EIGENVALUES*/13X,*REAL*,9X,*IMAGINARY*,
     1 6X,*NATURAL FREQ.*,5X,*DAMPING RATIO*)
      DO 2868 I=1,NPOLES
      J=IBLNK
      IF(EVI(I).GT.0.) J=IPOM
 2868 WRITE(6,2869)I,EVR(I),J,EVI(I),WN(I),DAMPR(I)
 2869 FORMAT(3X,I3,3X,G12.6,2X,A2,G12.6,4X,2G16.6)
      WRITE(6,2871)
2871  FORMAT(/////)
      RETURN
C                  *** CALCULATE EIGENVALUE SENSITIVITY ***
      ENTRY ESBTCH
C ==========  DETERMINE EIGEN PARAMETER CODE
      CALL CODGEN(ESPAR,0,INDEP),RETURNS(4000)
C =========  GET NOMINAL VALUE OF EIGEN PARAMETER
      CALL VAROUT(INDEP,PARAMO)
      IF(PARAMO.NE.0.) GO TO 2950
C ========  DEFAULT PERTURBATION = .1 IF NOMINAL VALUE OF EIGEN
C           PARAMETER = 0
      DELTA=.1
      GO TO 2955
C =========  PERTURB EIGEN PARAMETER BY 5 OF NOMINAL
 2950 DELTA=PARAMO*.05
C =========  CALC. NOMINAL STABILITY MATRIX AND EIGENVALUES
 2955 CALL STABMX(NSIM,XDOTO,ICOUNT,RATIO,A,NLIN,0)
      CALL EGVL3(A,RATIO,EVR,EVI,IA,IB,IC,ID,DWORK,1.E-14,NLIN,NLIN)
C =========  ORDER EIGENVALUES TO DECREASING REAL PARTS
      CALL EVORDR(EVR,EVI,ID,NLIN)
C =========  PERTURB EIGEN PARAMETER
 2980 PARAM=PARAMO+    DELTA
      CALL VARMOD( INDEP,PARAM)
C ========  CALC. PERTURBED STABILITY MATRIX AND EIGENVALUES
      CALL STABMX(NSIM,XDOTO,ICOUNT,RATIO,A,NLIN,0)
      CALL EGVL3(A,RATIO,EVRP,EVIP,IAP,IBP,IC,ID,DWORK,1.E-14,NLIN,NLIN)
```

```
      CALL EVORDR(EVRP,EVIP,ID,NLIN)
      DELTA=ABS(PARAMO/DELTA)
C ===========  CALC. EIGEN SENSITIVITY
      DO 3000 I=1,NLIN
      RATIO(I)=EVR(I)
      RATIO(INDEX2(I,2,NLIN))=EVI(I)
      RATIO(INDEX2(I,5,NLIN))=EVRP(I)
      RATIO(INDEX2(I,6,NLIN))=EVIP(I)
      IF(EVR(I).EQ.0.) GO TO 2991
C =========  CALC. SENSITIVITY OF REAL PARTS
      RATIO(INDEX2(I,3,NLIN))=DELTA*(1.-EVRP(I)/EVR(I))
      GO TO 2993
 2991 IF(EVRP(I).EQ.0.) GO TO 2992
      RATIO(INDEX2(I,3,NLIN))=1.E36
      GO TO 2993
 2992 RATIO(INDEX2(I,3,NLIN))=0.
 2993 IF(EVI(I).EQ.0.) GO TO 2994
C ===========  CALC. SENSITIVITY OF IMAG. PARTS
      RATIO(INDEX2(I,4,NLIN))=DELTA*(1.-EVIP(I)/EVI(I))
      GO TO 3000
 2994 IF(EVIP(I).EQ.0.) GO TO 2995
      RATIO(INDEX2(I,4,NLIN))=1.E36
      GO TO 3000
 2995 RATIO(INDEX2(I,4,NLIN))=0.
 3000 CONTINUE
      WRITE(6,3020) ESPAR,PARAMO,PARAM
 3020 FORMAT(1H1 /20X,7H/*/*/*/,3X,*EIGENVALUE SENSITIVITY*,3X,7H/*/*/*/
     1 ,3X,A8,  * PERTURBED FROM *,G12.6,* TO *,G12.6///13X,*NOMINAL EIG
     2ENVALUES*,11X,*SENSITIVITY MEASURE*,10X,*PERTURBED EIGENVALUES*/
     3,2X,3(10X,*REAL*,7X,*IMAGINARY*)/)
C ========   PRINT RESULTS
      WRITE(6,3025)(I,(RATIO(I+(J-1)*NLIN),J=1,6),I=1,NLIN)
 3025 FORMAT(3X,I3,6G15.6)
      CALL VARMOD( INDEP,PARAMO)
      WRITE(6,2871)
      RETURN
 4000 WRITE(6,4001) ESPAR
 4001 FORMAT(//10X,31H*** WARNING ***  CAN*T IDENTIFY,1X,A8,1X,
     1 43HAS A VALID EIGENVALUE SENSITIVITY PARAMETER//)
      WRITE(6,2871)
      RETURN
      END
```

```
CLPRINT
      SUBROUTINE LPRINT(IPRINT,TIME)
C  VERSION 3.                          REVISED: MAY 5 1976
C  PURPOSE:  PROVIDE GENERAL LINEPRINTER OUTPUTS.
C  CALL SEQUENCE:  IPRINT - PRINT CONTROL VARIABLE.
C                  TIME   - CURRENT TIME.
C    IPRINT VALUE    QUANTITIES PRINTED
C       0 OR 1       STATES, RATES, AND TIME
C          2         STATES, RATES, VARIABLES, AND TIME
C          3         STATES, RATES, VARIABLES, (PARAMETERS AT TIME=0 ONLY)
C          4         STATES, RATES, VARIABLES, PARAMETERS, AND TIME
C          5         VARIABLES SPECIFIED IN  PRTNAM ARRAY
      COMMON/CNAMEX/NAMEX(1)/CNAMER/NAMER(1)/CNAMEV/NAMEV(1)
      COMMON/CNAMEP/NAMEP(1)
      COMMON/CX/X(1)/CXDOT/XDOT(1)/CV/V(1)/CP/P(1)
      COMMON/CORDER/NOX,NOV,NOP
      COMMON/CPRINT/PRTNAM(10),LPRT(10)/CDIFS/JSTART,KINIT,TP
      DIMENSION OUTPUT(10)
C ------>       TEST FOR LIST OPTION
      IF(IPRINT.EQ.5)GO TO 300
C ------>       PRINT STATES
      WRITE(6,11)TIME,(I,NAMEX(I),X(I),I=1,NOX)
11    FORMAT(//10X,*TIME = *,G10.4,30X,*STATES*/5(I4,1X,A8,2H= ,G11.5))
C ------>       PRINT RATES.
      WRITE(6,13)(I,NAMER(I),XDOT(I),I=1,NOX)
13    FORMAT(/57X,*RATES*/5(I4,1X,A8,2H= ,G11.5))
C ------>       TEST FOR VARIABLES OPTION.
      IF(IPRINT.LE.1)RETURN
C ------>       PRINT VARIABLES.
      WRITE(6,15)(I,NAMEV(I),V(I),I=1,NOV)
15    FORMAT(/57X,*VARIABLES*/5(I4,1X,A8,2H= ,G11.5))
C ------         TEST FOR PARAMETER PRINT OPTIONS
      IF(IPRINT.LE.2)RETURN
      IF(IPRINT.LE.3.AND.TIME.GT.0.)RETURN
      WRITE(6,17)(I,NAMEP(I),P(I),I=1,NOP)
17    FORMAT(/57X,*PARAMETERS*/5(I4,1X,A8,2H= ,G11.5))
      RETURN
C ------>       SCAN CODES AND GET CURRENT VALUES.
300   N=0
      DO 320 I=1,10
C ------>       TEST FOR LAST VARIABLE
      IF(LPRT(I).EQ.-1)GO TO 310
      CALL VAROUT(LPRT(I),OUTPUT(I))
      N=I
      GO TO 320
310   OUTPUT(I)=0.
320   CONTINUE
C ------>       TEST FOR NO LIST QUANTITIES IDENTIFIED
      IF(N.LT.1)RETURN
C ------>       PRINT HEADING WHEN KINIT = 0.
      IF(KINIT.EQ.0)WRITE(6,343)(PRTNAM(I),I=1,N)
343   FORMAT(/4X,*TIME*,3X,10(3X,A8,1X))
C ------>       PRINT LIST VALUES.
360   WRITE(6,363)TIME,(OUTPUT(I),I=1,N)
363   FORMAT(1X,G10.4,10G12.5)
      RETURN
      END
```

```
CLUEQS
      SUBROUTINE LUEQS(A,B,C,IA,NA,NB,MA,MB,MC,FPZ,IERROR)
C  VERSION 2.0                          REVISED: OCT 6 1975
C     SOLVES FOR B WHERE A*B=C
C     INPUTS ARE;
C        A     THE MULTIPLIER MATRIX
C        C     THE RIGHT HAND SIDE OF THE EQUATION
C        NA    THE ORDER OF THE MATRIX A
C        NB    THE NUMBER OF COLUMNS OF MATRICES B AND C
C        MA    THE ROW DIMENSION OF A
C        MB    THE ROW DIMENSION OF B
C        MC    THE ROW DIMENSION OF C
C        FPZ   THE PRECISION INDICATOR
C
C     ON SUCCESSFUL COMPLETION (IERROR=0)
C        B    THE SOLUTION MATRIX
C        A    THE LU DECOMPOSITION OF A
C        IA   VECTOR OF ROW PERMUTATIONS OF THE LU DECOMPOSITION IN A
C
C     DIMENSION INFORMATION
C        IA    VECTOR OF LENGTH NA
C        MB AND MC MUST BE AT LEAST NA
C
C     NOTES;
C        IF NB IS 0, ONLY THE LU DECOMPOSITION OF A IS COMPUTED.
C
C     THIS PROGRAM WAS DESIGNED AND CODED BY A. FREDERICK FATH OF
C     BOEING COMPUTER SERVICES, SEATTLE, WASHINGTON.  THIS VERSION
C     WAS COMPLETED DURING APRIL 1975.
C
      DIMENSION A(MA,1),B(MB,1),C(MC,1),IA(1)
      ERR=FPZ*50.
      IERROR=0
      DO 10 I=1,NA
      IA(I)=I
10    CONTINUE
      NA1=NA-1
      IF(NA.LE.1) GO TO 46
      DO 45 I=1,NA1
      I1=I+1
      I2=I-1
C     SEARCH FOR PIVOT
      SM=0.
      DO 18 J=I,NA
      IF(ABS(A(J,I)).LT.SM) GO TO 18
      IR=J
      SM=ABS(A(J,I))
 18   CONTINUE
C     INTERCHANGE ROWS
      IF(I.EQ.IR) GO TO 23
      DO 20 J=1,NA
      SM=A(I,J)
      A(I,J)=A(IR,J)
      A(IR,J)=SM
20    CONTINUE
      J=IA(I)
      IA(I)=IA(IR)
      IA(IR)=J
```

```
C       COMPUTE COLUMN OF L
 23     IF(A(I,I))24,100,24
 24     A(I,I)=1./A(I,I)
        DO 25 J=II,NA
        A(J,I)=A(J,I)*A(I,I)
25      CONTINUE
C       COMPUTE ROW OF U
        IF(I.EQ.1) GO TO 35
        DO 30 J=II,NA
        SUM=A(I,J)
        SB=ABS(SUM)
        DO 28 K=1,I2
        SA=A(I,K)*A(K,J)
        IF(ABS(SA).GT.SB) SB=ABS(SA)
        SUM=SUM-SA
28      CONTINUE
        IF(ABS(SUM).LT.ERR*SB) SUM=0.
 30     A(I,J)=SUM
C       COMPUTE UNNORMALIZED COLUMN OF L
 35     DO 45 J=II,NA
        SUM=A(J,II)
        SB=ABS(SUM)
        DO 38 K=1,I
        SA=A(J,K)*A(K,II)
        IF(ABS(SA).GT.SB) SB=ABS(SA)
        SUM=SUM-SA
38      CONTINUE
        IF(ABS(SUM).LT.ERR*SB) SUM=0.
        A(J,II)=SUM
45      CONTINUE
 46     IF(A(NA,NA))47,100,47
 47     A(NA,NA)=1./A(NA,NA)
        CALL SLVEQ(A,B,C,IA,NA,NB,MA,MB,MC,FPZ,IERROR)
        RETURN
 100    WRITE(6,101)
 101    FORMAT(27H0  ** MATRIX IS SINGULAR **)
        IERROR=1
        RETURN
        END
```

```
CNAMES
      SUBROUTINE NAMES(IPHRS,NAME,NUNIT,NO,ITNO,MODE)
C  PURPOSE:  LOADS ALPHANUMERIC NAMES OF QUANTITIES IDENTIFIED BY
C            DEFINE STATEMENTS.
C  CALL SEQUENCE:  IPHRS  = ARRAY CONTAINING NEXT PHRASE TO BE EXAMINED.
C                  NAME   = ARRAY TO BE LOADED WITH NAMES OF
C                           DEFINED QUANTITIES.
C                  NUNIT  = ARRAY, TO BE LOADED WITH UNIT NAMES
C                           OF DEFINED QUANTITIES.
C                  NO     = NUMBER OF DEFINED QUANTITIES.
C                  ITNO   = POSITION OF GIVEN QUANTITY IN NAME ARRAY.
C                  MODE   = MODE OF OPERATION INDICATOR.
C                      MODE = 0 WHEN ITNO HAS BEEN LOADED.
C                      MODE = 1 WHEN NAME HAS BEEN LOADED.
      DIMENSION NAME(NO),NUNIT(NO)
      REAL IPHRS,NAME,NUNIT
C  TEST FOR NUMERIC FIRST CHARACTER.
      CALL NUMERC(IPHRS),RETURNS(100)
      GO TO 200
C  TEST THAT ITNO IS WITHIN ALLOWABLE RANGE.
100   IF(ITNO.LT.1.OR.ITNO.GT.NO) GO TO 120
      IF(MODE.NE.0) GO TO 110
C  LOAD NAME
      NAME(ITNO)=IPHRS
      MODE=1
      RETURN
C  LOAD UNITS NAME.    (ALL NAMES WILL BE PUT IN WORD 1 FOR NOW.)
110   NUNIT(1    )=IPHRS
      RETURN
120   WRITE(6,121) ITNO,IPHRS
121   FORMAT(15X,15H*** WARNING ***,I8,40H EXCEEDS THE ALLOWABLE INDEX R
     1ANGE FOR  ,A10,34H THIS QUANTITY WILL NOT BE DEFINED)
      RETURN
C  CONVERT IPHRS TO I FORMAT.
200   CALL BCDREL(FLNO,IPHRS)
      ITNO=FLNO
      MODE=0
      RETURN
      END
```

```
CNATFRQ
      SUBROUTINE NATFRQ(EVR,EVI,WN,DAMPR,NLIN,NPOLES)
C  PURPOSE:  TO ORDER EIGENVALUES WITH MOST POSITIVE REAL PARTS FIRST,
C            CALCULATE NATURAL FREQUENCIES AND DAMPING RATIOS,
C            AND TO ELLIMINATE THE ROOTS WITH NEGATIVE IMAGINARY PARTS.
C  CALL SEQUENCE:  EVR = REAL PARTS OF EIGENVALUES (NLIN ARRAY ON ENTRY
C                        NPOLES ARRAY UPON RETURN. )
C                  EVI = IMAGINARY PARTS OF EIGENVALUES
C                  WN = NATURAL FREQUENCY ARRAY
C                  DAMPR = DAMPING RATIO ARRAY.
C                  NLIN = SYSTEM ORDER.
C                  NPOLES = NUMBER OF POLES WITH IMAGINARY PARTS >= 0.
      DIMENSION EVR(1),EVI(1),WN(1),DAMPR(1)
      NPOLES=0
C  ELLIMINATE ALL POLES WITH NEGATIVE IMAGINARY PARTS.
      DO 50 I=1,NLIN
      IF(EVI(I).LT.0) GO TO 50
      NPOLES=NPOLES+1
      EVR(NPOLES)=EVR(I)
      EVI(NPOLES)=EVI(I)
   50 CONTINUE
C  SORT REAL PARTS OF POLES INTO ASCENDING ORDER.
      CALL FSHELL(EVR,WN,NPOLES)
C  SORT IMAGINARY PARTS TO CORRESPOND TO REAL PARTS.
      CALL SHELLX(EVI,WN,NPOLES)
C  REVERSE EIGENVALUE SEQUENCE.  (PUT INTO DECENDING ORDER)
      NLIN1=NPOLES+1
      NLIN2=NPOLES/2
      DO 100 I=1,NLIN2
      I2=NLIN1-I
      EVRS=EVR(I)
      EVIS=EVI(I)
      EVR(I)=EVR(I2)
      EVI(I)=EVI(I2)
      EVR(I2)=EVRS
  100 EVI(I2)=EVIS
C  CALCULATE NATURAL FREQUENCIES AND DAMPING RATIOS.
      DO 200 I=1,NPOLES
      WN(I)=SQRT(EVR(I)*EVR(I)+EVI(I)*EVI(I))
      DAMPR(I)=0.
      IF(WN(I).NE.0.)DAMPR(I)=-EVR(I)/WN(I)
  200 CONTINUE
      RETURN
      END
```

```
CNONSIM
      OVERLAY(NONSIM,0,0)
      PROGRAM NONSIM(INPUT=100,OUTPUT=200,TAPE5=INPUT,TAPE6=OUTPUT,
     1 TAPE1=600, PUNCH=100,TAPE3=PUNCH,TAPE30=1000,TAPE25=1000)
C  VERSION 3.                          REVISED: APRIL 30 1976
C  PURPOSE:  MAIN PROGRAM FOR THE BATCH VERSION OF NONSIM.
      COMMON/COVRLY/INST,LOKSS,LOKSIM,CPUSEC
      COMMON/CPROV/XMIN1,XMAX1,XMIN2,DELTA2,CURVES,PRINT,PRATE,OUTRAT,
     1 AMODE,TINC,TMAX,FMAX,FMIN,XSTART,XSTOP,SPOINT,SSLIM,RSTART,RSTOP,
     2 RPOINT,RLMIN,RLMAX,IMMIN,IMMAX,OCMOD,OCORD,TZERO
      COMMON/CPRON/DEPEN,INDEP1,INDEP2,ESPAR,NINPUT,NOUT,INDEP,RLPAR
      COMMON/CSMPAR/SMPAR(10),ICIND(2)
      COMMON/CORDER/NSIM,NOV,NOP/CWORKN/NN,N(7)
      COMMON/CSIMUL/IPRIN,IPRATE,IOUT,NPTS,NPTMAX,INDMAX,TINC2,TMAX2,
     1 INDEX,IPLOT,IDENT(4)
      REAL IDENT
      EQUIVALENCE(NSIM,NOX)
      COMMON /CPLOTS/ INDPLT,INDWR,IOPT(30),PLOTID( 5),PTITLE( 8),
     +                IPOPT(10)
      REAL    SMPAR,DEPEN,INDEP1,INDEP2,ESPAR,NINPUT,NOUT,INDEP,RLPAR
C  CALL USER FURNISHED INPUT ROUTINE.
      CALL DATAIN
      CALL OVERLAY(4HINIT,1,0)
      INST=1
C  INTERPRETATION ROUTINE TO READ INSTRUCTIONS.
100   CALL OVERLAY(6HINTERP,2,0,6HRECALL)
      IF(INST.LE.0) STOP
C  BRANCH TO SPECIFIED ANALYSIS.
      GO TO (100,100,100,100,100,100,100,100,100,100,
     1       100,100,200,200,100,100,100,100,100,100,
     2       100,100,100,100,100,300,400,400,500,600,
     3       700,800,420,100,800,100,100,100,100,100,
     4       100,100,100,100,100,100,100,100,100,100,
     5       100,100,100,100,1000,100,100)              ,INST
C  GENERAL FUNCTION OF ONE INDEPENDENT VARIABLE.
200   CALL OVERLAY(6HGFBTCH,3,0)
      GO TO 100
  300 IF(LOKSIM.EQ.1) GO TO 310
      WRITE(6,301)
301   FORMAT(//15X,15H*** WARNING ***,5X,*SIMULATION WILL NOT BE RUN DUE
     1 TO FAILURE TO REACH VALID STEADY STATE*//)
      GO TO 100
  310 IPRIN=PRINT
      IPRATE=PRATE
      IOUT=OUTRAT
      TINC2=TINC
      TMAX2=TMAX
      CALL OVERLAY(6HSIBTCH,4,0)
      LOKSS=1
      GO TO 100
400   CALL OVERLAY(6HLABTCH,5,0)
      GO TO 100
420   CONTINUE
      GO TO 100
500   CALL OVERLAY(6HSMBTCH,6,0)
      GO TO 100
600   CALL OVERLAY(6HTFBTCH,7,0)
      GO TO 100
```

```
700     CALL OVERLAY(6HSSBTCH,10B,0)
        GO TO 100
800     CALL OVERLAY(6HRLBTCH,11B,0)
        GO TO 100
C ==================  DESIGN O.C.  ==================
C ------------------------------  GENERATE LINEAR SYSTEM MODEL  -- PROGRAM O
1000    CALL OVERLAY(6HNONSIM,12B,0)
C ------------------------------  GENERATE OPTIMAL CONTROLLER  -- PROGRAM OC
C       CALL OVERLAY(6HNONSIM,13B,0)
        GO TO 100
        END
```

```
CNRKV
      OVERLAY(NONSIM,4,1)
      PROGRAM NRKV
C   PURPOSE:  PERFORM INTERATION USING RUNGE-KUTTA ALGORITHIM
C   VERSION 1.                          REVISED: JULY 11 1975
      COMMON/CX/X(1)/CXDOT/XDOT(1)/CTIME/TIME/CORDER/NSIM,NOV,NOP
      COMMON/CWORK/A(1)/CWORKN/NN,N1,N2,N3,N4,N5,N6,N7
      COMMON/CNTRLS/ANTYPE,IPRINT,IMODE,ERROR(1)
      COMMON/CSIMUL/IPRIN,IPRATE,IOUT,NPTS,NPTMAX,INDMAX,TINC,TMAX
      COMMON/NRKVS3/IDIAG,AHMINM,AHSIRT,INTFLG,MAXN,MFAIL1,MSTEP2,ISTABL
      COMMON/CDIFS/JSTART,KINIT,TP
      IF(KINIT.EQ.0) CALL RKINIT
      CALL NRKVS(TP,NSIM,A(NN),A(N1),A(N2),A(N3),A(N4),A(N5),A(N6),
     1A(N7))
      END
```

```
CNRKVS
      SUBROUTINE NRKVS(TP,NV,XX,XS,DS,Z,ZZ,DHIST,XHIST,ECNT)
C  VERSION 2.1                              REVISED: MARCH 23 1976
      COMMON/CX/X(1)/CXDOT/XDOT(1)/CTIME/TIME/CORDER/NSIM,NOV,NOP
      COMMON/CNTRLS/ANTYPE,IPRINT,IMODE,ERROR(1)
      COMMON/CSIMUL/IPRIN,IPRATE,IOUT,NPTS,NPTMAX,INDMAX,TINC,TMAX
      COMMON/NRKVS3/IDIAG,AHMINM,AHSTRT,INTFLG,MAXN,MFAIL1,MSTEP2,ISTABL
      COMMON/COIFS/JSTART,KINIT/ERMESS/IFATAL,IERR
      DIMENSION XX(1),XS(1),DS(1),Z(1),ZZ(1),DHIST(1),XHIST(1),ECNT(1)
      LOGICAL REJECT,RETRN
      DATA XRHO,DRHO/.99,.75/,A,B,C/.45660211,3.2761459,2.7327480/
      ERR=1.E-5
C
C     ***************************************************************
C
C     INITIALIZATION FOR FIRST CALL TO THIS PROGRAM.
C         ASSIGNMENTS ARE MADE SO AS
C         TO USE EITHER SIMPSON'S RULE OR RUNGE KUTTA FORMULAS
C
C               CHECK FOR NV EXCEEDING DIMENSION SIZE
C
      RETRN = .FALSE.
      ITERCT=0
      IF(KINIT.GT.0) GO TO 200
      KINIT = 1
      REJECT = .FALSE.
  108 DO 120 I=1,NV
      DHIST(I) = 0.0
      ECNT(I)=0.
      XX(I) = X(I)
      XHIST(I) = AMAX1(ABS(X(I)),ERROR(I))
      IF (XHIST(I)) 110,110,120
  110 XHIST(I) = 1.0
  120 CONTINUE
      TS=TIME
      TR=TP-TS
      IF(TR)11,10,11
   10 CALL EQMO(TIME,TINC,0)
      RETURN
   11 CONTINUE
C****
C     SET STABILITY CONTROLS
      AHMAX=1.E+37
   13 JLMAX=100
      TSTBL =4.
      TSTBL2=2.
      RTDN  =.75
      RTUP1 =.25
      RTUP2 =.75
      JDELAY=5
      JHIST =0
   14 JLOCAL=0
C****
C     INITIAL STEP AND MINIMUM STEP - SET BY USER OR PROGRAM
      IF(AHSTRT)122,122,121
  121 AHS = AHSTRT
      GO TO 123
  122 AHS =.01*ABS(TP-TS)
```

```
  123 IF(AHMINNM)125,125,124
  124 AHMIN = AHMINNM
      GO TO 200
125   AHMIN=AMIN1(1.E-5,TINC/10000.)
C****
C
C     ***************************************************************
C
C     INITIALIZATION FOR SUBSEQUENT CALLS TO THIS PROGRAM
C         CONTINUATION OF AN INTEGRATION AFTER A NEW
C         PRINT TIME (TP) IS SET
C****
C     EVALUATE DERIVATIVES AT STARTING POINT
C
  200 ENDT = TP
      CALL EQMO(TIME,TINC,0)
C****
      DO 210 I=1,NV
      XX(I)=X(I)
  210 DS(I) = XDOT(I)
      TR = TP-TS
      IREJEC = 0
      TTEST = TS
      TEST= .005*ABS(TR)
      NSTEP = 0
      NSTEP2= 0
      NFAIL1= 0
C
C     ***************************************************************
C
C     RUNGE KUTTA FORMULAS
C         A ONE STEP,FOLLOWED BY A TWO STEP,INTEGRATION IS PERFORMED.
C         THE INCREMENTS TO THE DEPENDENT VARIABLES ARE FOUND IN Z,ZZ
C****
C                   DO ONE STEP OF LENGTH H
  300 AH =AMIN1(AHS,ABS(TR))
      H=SIGN(AH,TR)
      H2= .5*H
      H6= H2/3.
      TIME = TS+ H2
      DO 302 I=1,NV
      XS(I)=XX(I)
  302 X(I)=XS(I)+H2*DS(I)
      CALL EQMO(TIME,TINC,0)
      DO 304 I=1,NV
      Z(I)= DS(I)+2.*XDOT(I)
  304 X(I)= XS(I)+H2*XDOT(I)
      CALL EQMO(TIME,TINC,0)
      TIME =TS +H
      DO 306 I=1,NV
      Z(I)= Z(I)+ 2.*XDOT(I)
  306 X(I)= XS(I)+H*XDOT(I)
      CALL EQMO(TIME,TINC,0)
      H4=.25*H
      H12=H4/3.
      TIME=TS+ H4
      DO 310 I=1,NV
      Z(I)=H6*(Z(I)+XDOT(I))
```

```
C****
C
C****
C                    DO FIRST HALF STEP OF LENGTH H/2
  310 X(I)=XS(I)+H4*DS(I)
      CALL EQMO(TIME,TINC,0)
      DO 314 I=1,NV
      ZZ(I)=DS(I)+2.*XDOT(I)
  314 X(I)= XS(I)+H4*XDOT(I)
      CALL EQMO(TIME,TINC,0)
      TIME=TS+H2
      DO 316 I=1,NV
      ZZ(I)=ZZ(I)+2.*XDOT(I)
  316 X(I)= XS(I)+H2*XDOT(I)
      CALL EQMO(TIME,TINC,0)
C****
C
C****
C                    SET UP SECOND HALF STEP
C
      DO 320 I=1,NV
      ZZ(I)=ZZ(I)+XDOT(I)
      X(I)= XS(I)+H12*ZZ(I)
  320 XS(I)=X(I)
      CALL EQMO(TIME,TINC,0)
C****
C
C                    DO SECOND HALF STEP OF LENGTH H/2
C
C****
      TIME=TIME+H4
      DO 322 I=1,NV
      X(I)=XS(I)+H4*XDOT(I)
  322 ZZ(I)=ZZ(I)+XDOT(I)
      CALL EQMO(TIME,TINC,0)
      DO 324 I=1,NV
      ZZ(I)=ZZ(I)+2.*XDOT(I)
  324 X(I)= XS(I)+H4*XDOT(I)
      CALL EQMO(TIME,TINC,0)
      TIME=TS+H
      DO 326 I=1,NV
      ZZ(I)=ZZ(I)+2.*XDOT(I)
  326 X(I)= XS(I)+H2*XDOT(I)
      CALL EQMO(TIME,TINC,0)
      DO 330 I=1,NV
  330 ZZ(I)=H12*(ZZ(I)+XDOT(I))
C****
C                    NOW GO TO ERROR CHECKING SEQUENCE
C     ********************************************************************
C
C     ERROR CHECKING SEQUENCE.
C          IN THIS SECTION,THE SIZE STANDARD IS UPDATED,THE ERRORS ARE
C          COMPUTED,THE WORST ERROR IS COMPUTED,AND DECISIONS ARE MADE
C          AS TO HOW TO PROCEED.
  500 TERR= 0.
      RECIP = 1./AH
      IERCT=0
      DO 508 I=1,NV
```

```
      CHNG=AMAX1(ABS(Z(I)),ABS(ZZ(I)))
      Z(I)=(ZZ(I)-Z(I))/15.
      IF(CHNG-ERROR(I)) 504,504,506
  504 DHIST(I)= 0.
      GO TO 508
  506 DHIST(I)= AMAX1(CHNG*RECIP,DRHO*DHIST(I))
C****
C          SIZE STANDARD
      ERRSTD=AMAX1(DHIST(I)*AH,XHIST(I))
C****
C          ERROR FOR EACH EQUATION
      RERR=ABS(Z(I)/(ERRSTD*ERR))
      RERR=AMIN1(RERR,Z(I)*50./ERROR(I))
C****
C          WORST ERROR FOR ALL EQUATIONS
      IF(TERR.GT.RERR) GO TO 508
      TERR=RERR
      IERCT=I
  508 CONTINUE
C****
C
C****
C                  NOW PROCESS ITEST TO DECIDE WHAT TO DO WITH THIS
C                  INTEGRATION CYCLE
C
C                                 IF TERR.LE.1 THEN THIS STEP IS SMALL
      IF(TERR.LE.1.) GO TO 512
C                                    REDUCE JHIST
      IF(JHIST) 511,510,509
  509 JHIST=JHIST-1
C                                 IF TERR.GT.TSTBL THEN INVOKE STABILITY
  510 IF(TERR.LT.TSTBL) GO TO 519
C                                    ONLY IF PREVIOUS STEP WAS SMOOTH
      IF(JLOCAL.EQ.0) GO TO 511
C****
C                                     WE ASSUME WE HAVE EXCEEDED THE
C                                     RANGE OF STABILITY. TIGHTEN CONTROLS
C
      AHMAX =AH*RTDN
      TSTBL =1.+.5*TSTBL
      RTDN  =.75*RTDN+.2421875
      RTUP1 =.5*RTUP1
      RTUP2 =1.-RTUP1
      JLOCAL=0
      JHIST =0
      IF(JDELAY.LT.20) JDELAY=JDELAY+2
C****
C                                    CHECK FOR REJECTION
  511 IF(TERR-10.) 520,520,515
C****
C                                    STEP ACCEPTED  -  NEXT STEP INCREASED
  512 IF( ABS(TR)-AH) 521,521,513
  513 IF(REJECT) GO TO 600
      AHS=AMIN1(2.0,(1.+C*TERR)/(A+B*TERR))*AH
C                                 INCREASE JLOCAL AND JHIST (BOUNDED)
      IF(JLOCAL.GT.JLMAX) GO TO 514
      JLOCAL=JLOCAL+1
      JHIST =JHIST+1
```

```
C                                 BOUND AHS
  514 IF(AHMAX.GE.AHS) GO TO 600
C                                 CHECK FOR AHMAX TO INCREASE
      TEMP = AHS
      AHS   =AHMAX
      IF(JLOCAL.LT.JDELAY) GO TO 600
C                                 INCREASE AHMAX  USING RTUP PARAMETERS
      AHMAX =TEMP*RTUP1+AHMAX*RTUP2
      JLOCAL=0
      GO TO 600
C****
C                                       STEP REJECTED
  515 REJECT = .TRUE.
      ECNT(IERCT)=ECNT(IERCT)+1
      IREJEC = IREJEC +1
      IF(AHS.GT.AHMIN) GO TO 518
      NFAIL1 =NFAIL1 +1
      IF(NFAIL1.GT.1) GO TO 516
      TF1=TS
  516 TF2=TS
      GO TO 700
  518 AHS=AMAX1(AHMIN,AMAX1(.5,(TERR*A+B)/(TERR+C))*AH)
      GO TO 300
C****
C                                       STEP ACCEPTED  -  NEXT STEP DECREASED
C                                 IF TERR.GT.2 ERROR IS MODERATELY SEVERE
  519 IF(TERR.LT.TSTBL2) GO TO 520
      IF(JDELAY.LT.20) JDELAY=JDELAY+1
  520 AHS=AMAX1(AHMIN,((TERR*A+B)/(TERR+C))*AH)
      IF(JLOCAL.EQ.0) GO TO 522
      JLOCAL=0
C                                 CHECK FOR AHMAX TO DECREASE
      IF(JDELAY.LE.10) GO TO 522
      AHMAX =AHMAX*RTDN+AHS*(1.-RTDN)
      RTUP1 =RTUP1*.875
      RTUP2 =1.-RTUP1
      JHIST =0
  522 IF( ABS(TR)-AH) 521,521,600
  521 RETRN =.TRUE.
C****
C
C     ************************************************************************
C     SYSTEM UPDATE SECTION.
C         AT THIS POINT A STEP HAS BEEN ACCEPTED,UPDATE THE SYSTEM
C         VARIABLES AND PREPARE FOR A NEW STEP UNLESS TP HAS BEEN
C         REACHED OR A DIAGNOSTIC MESSAGE IS REQUIRED.
  600 REJECT =.FALSE.
  610 DO 620 I=1,NV
      Z(I)=ZZ(I)+Z(I)
      X(I)=Z(I)+XX(I)
      XX(I) = X(I)
  620 XHIST(I)= AMAX1(ABS(X(I)),XRHO*XHIST(I))
      TIME=TS+H
      TS=TIME
C ----------  TURN ON ERROR MESSAGES IN MODEL
  622 IERR = 1
      CALL EQMO(TIME,TINC,0)
C ----------  TURN OFF ERROR MESSAGES IN MODEL
```

```
      IERR = 0
C*** IF GEAR INTEGRATOR MODE IS SPECIFIED RETURN AFTER 100 STEPS
      ITERCT=ITERCT+1
      IF(IMODE.EQ.1.AND.ITERCT.GT.100) RETURN
  624 DO 630 I=1,NV
  630 DS(I)=XDOT(I)
      NSTEP=NSTEP+1
C
C                                CHECK FOR SMOOTH OPERATION
      IF(JHIST.LT.20) GO TO 633
C                                     YES - LOOSEN STABILITY CONTROLS
      JHIST =0
      TSTBL =.5*TSTBL+2.
      RTUP2 =.5*RTUP2+.375
      RTUP1 =1.-RTUP2
      RTDN  =.75*RTDN+.1875
      IF(JDELAY.GT.5) JDELAY=JDELAY-2
C
  633 IF(IDIAG.EQ.0) GO TO 636
      IF(NSTEP.LE.MAXN)   GO TO 730
  636 IF(TP-ENDT)638,640,638
  638 TP=ENDT
      IF(ENDT.EQ.TS) RETURN
      RETRN = .FALSE.
  640 TR= TP- TS
      IREJEC = 0
      IF(RETRN) RETURN
      NSTEP2 = NSTEP2 + 1
      IF(NSTEP2.LT.MSTEP2) GO TO 300
      IF (ABS(TS-TTEST)-TEST) 710,710,644
  644 TTEST =TS
      NSTEP2= 0
      GO TO 300
C
C     ***************************************************************
C
C     DIAGNOSTIC SECTION.
C         THREE LEVELS OF DIAGNOSTICS ARE POSSIBLE, TWO OF
C         WHICH CAUSE AN ERROR RETURN
C
C****
C
C                  1.IF AN ERROR TEST IS NOT SATISFIED AT MINIMUM STEP,
C                    A MESSAGE IS PRINTED. THIS IS ALLOWED TO HAPPEN
C                    AT MOST MFAIL1 TIMES AT WHICH TIME
C                    THE ERROR RETURN IS MADE
C
  700 WRITE(6,702) TS,RERR
  702 FORMAT(1H0,23HERROR CHECK FAILURE AT , E14.8,10X,23HERROR INDICATO
     1RS FOLLOW,/,(1H F9.3,9F9.3))
      IF(NFAIL1.LE.MFAIL1) GO TO 610
      WRITE(6,704)
  704 FORMAT(1H0,51HEXCESSIVE ERROR CHECK FAILURES AT MINIMUM STEP SIZE)
      WRITE(6,706) TS,(X(I),I=1,NV)
  706 FORMAT(1H0,26HINTEGRATION TERMINATED AT ,E14.8,
     150H WITH THE FOLLOWING DEPENDENT VARIABLE EVALUATIONS,(/&E14.8))
      WRITE(6,708)TF1,TF2
  708 FORMAT(1H0,51HFIRST ERROR FAILURE AT MINIMUM STEP FOR INDEP VAR =,
```

```
     1E16.8/52H  LAST ERROR FAILURE AT MINIMUM STEP FOR INDEP VAR =,E16.
     28)
      GO TO 900
C
C****
C
C                    2. IF AN INTEGRATION TAKES MSTEP2 STEPS IN A SMALL
C                      FRACTION OF THE CURRENT PRINT INTERVAL (TEST IS
C                      THE VARIABLE SET TO THIS FRACTION) IT IS LIKELY
C                      THAT EXCESSIVE COMPUTER TIME WILL BE CONSUMED
C                      BEFORE THE PROBLEM IS COMPLETED,THEREFORE
C                      THE ERROR RETURN IS MADE
C
  710 WRITE (6,712) MSTEP2,TTEST,TS
  712 FORMAT(1HO,31HINTEGRATION PROCEDURE REQUIRED ,I3,6H STEPS/1HO,27HF
     1ROM INDEPENDENT VARIABLE =,E14.8/28H   TO INDEPENDENT VARIABLE =,E
     214.8/1HO,40HCOMPUTATION CONSIDERED PROHIBITIVLY SLOW)
      WRITE(6,706) TS,(X(I),I=1,NV)
      IF (NFAILI.NE.0) WRITE (6,708) TF1,TF2
      GO TO 900
C
C                    3. AS A DEBUGGING AID,A DIAGNOSTIC MAY BE PRINTED
C                      EACH STEP, HOWEVER,NO MORE THAN  MAXN  OF THESE
C                      CAN BE PRINTED IN EACH PRINT INTERVAL. THIS IS TO
C                      PREVENT THE INADVERTENT  GENERATION  OF
C                      EXCESSIVE OUTPUT.
C
  730 CONTINUE
      WRITE(6,742)NSTEP,IREJEC,TS,H,AHMAX,TERR,RERR
  742 FORMAT(1H ,2I4,4E14.5, 8F8.2,/,(1H 10X,15F8.2))
      GO TO 636
C     ***************************************************************
C
C     WHEN PROCEDURE STOPS BECAUSE OF ERROR, SET TIME TO TMAX + RETURN
  900 TIME=TMAX+1
      RETURN
      END
```

```
CPLINIT
      SUBROUTINE PLINIT
C
C     INITIALIZE FOR PLOTTING
C
      COMMON /CPLOTS/ INDPLT,INDWR,IOPT(30),PLOTID( 5),PTITLE( 8),
     +                IPOPT(10)
      COMMON /CSCALE/ SCALE(5,4,6),NVAR(5,2,6),NPLTS(6)
      DIMENSION DFLTID(5)
      DATA BLNK /10H          /
      DATA DFLTID /50H NONSIM PLOTS
     + /
C
      REWIND 30
      INDPLT = 0
      INDWR = 0
      DO 10 I=1,30
   10 IOPT(I) = 0
      DO 20 I=1,5
   20 PLOTID(I) = DFLTID(I)
      DO 30 I=1,8
   30 PTITLE(I) = BLNK
      DO 40 I=1,10
   40 IPOPT(I) = 0
      IPOPT(5) = 1
      DO 50 I=1,6
   50 NPLTS(I) = 0
      RETURN
      END
```

```
CPLOTAB
      SUBROUTINE PLOTAB(NTAB)
C  VERSION  1.                          REVISED  MAY 18 1976
C  PURPOSE:  PLOT TABULAR DATA
C  CALL SEQUENCE:    NTAB  -  I.D. NO. OF TABLE TO BE PLOTTED
C                          NTAB = -1   INDECATES ALL TABLES ARE TO BE PLOTTED
C  DESIGNED BY: J.D. BURROUGHS                    MAY 1976
      COMMON/CPLOTS/INDPLT,INDWR,IOPT(30),PLOTID(5),PTITLE(8),IPOPT(10)
      COMMON/CTABNA/TABNAM(1)/CMAXDI/NOTAB,MAXDIM(1)/CLOCTA/LOCTAB(1)
     1 /CTABLE/TABLES(1)
      REAL XOPT(30)
      EQUIVALENCE(XOPT(1),IOPT(1))
      DATA IBLNK/1OH          /
C ========  TEST IF ALL TABLES ARE TO BE PLOTTED
      IF(NTAB.EQ.-1)GO TO 100
C ----------  PLOT ONE TABLE
      II=NTAB
      N=NTAB
      GO TO 200
C ----------  PLOT ALL TABLES
100   II=1
      N=NOTAB
C ==========  SCAN TABLES TO BE PLOTTED
200   DO 4000 I=II,N
      IOPT(3)=IBLNK
      IOPT(4)=IBLNK
C ----------  GET DATE AND TIME
      CALL DTTIM(IOPT(3))
C ----------  GET STARTING LOCATION OF TABLE DATA
      LOC=LOCTAB(I)
C ----------  SELECT GENERAL FUNCTION PLOT OPTION
      IOPT(1)=1
C ----------  ADVANCE CASE NO. COUNTER
      IOPT(2)=IOPT(2)+1
C ----------  LOAD TABLE NAME
      XOPT(5)=TABNAM(I)
C ----------  PRIMARY INDEPENDENT NAME
      XOPT(6)=7HPRIMARY
C ----------  SECONDARY INDEPENDENT NAME
      XOPT(7)=7HSECOND
      DO 3100 J=8,13
3100  IOPT(J)=0
C ----------  NUMBER OF PRIMARY POINTS
      NX=TABLES(LOC+1)
      IOPT(14)=NX
C ----------  NUMBER OF SECONDARY POINTS
      NZ=TABLES(LOC+2)
      IF(NZ.LE.1)NZ=0
C ----------  LIMIT NO. OF SECONDARY POINTS TO 15 DUE TO IOPT DIMENSIONS
      NZLIM=MINO(NZ,15)
      IOPT(15)=NZLIM
C ==========  LOAD SECONDARY INDEPENDENT VARIABLE TABLE
      DO 3200 J=1,NZLIM
3200  XOPT(J+15)=TABLES(LOC+2+J)
      WRITE(30)IOPT,PLOTID,PTITLE
      LINDEP=LOC+NZ+2
C ==========  SCAN SECONDARY POINTS  ==========
      WRITE(30)((TABLES(LOC+J*NX+NZ+2+K),K=1,NX),(TABLES(LINDEP+K),
```

```
     1 K=1,NX),J=1,NZLIM)
4000  CONTINUE
      INDWR=1
      RETURN
      END
```

```
CPREC2
      SUBROUTINE PREC2(A,B,D,IA,IB,IC,ID,NSM,N,M)
C     SUBROUTINE TO PRECONDITION A MATRIX A BY REDUCING IT TO
C     UPPER BLOCK TRIANGULAR FORM AND SCALING
C     A IS N ORDER MATRIX,AND IS UNCHANGED BY PREC2
C     B IS N ORDER MATRIX WHICH CONTAINS PRECONDITIONED MATRIX
C     ON OUTPUT
C     D,IA,IB,IC, AND ID ARE WORK VECTORS OF DIMENSION GREATER THAN N
C     ON OUTPUT, IA CONTAINS THE ORDER OF THE VARIABLES, IB
C     CONTAINS THE NUMBER OF VARIABLES  IN EACH IRREDUCIBLE SUBBLOCK
C     NSM IS OUTPUT INDICATING NUMBER OF BLOCKS ON THE DIAGONAL.
C     IC CONTAINS LOCATIONS OF LAST ELEMENTS IN DIAGONAL SUBBLOCKS
C     D CONTAINS SCALE FACTORS ASSOCIATED WITH REARRANGED VARIABLES
C
C     THIS PROGRAM WAS DESIGNED AND CODED BY A. FREDERICK FATH OF
C     BOEING COMPUTER SERVICES, SEATTLE, WASHINGTON.  THIS VERSION
C     WAS COMPLETED DURING APRIL 1975.
C
      DIMENSION A(M,1),B(M,1),D(1),IA(1),IB(1),IC(1),ID(1)
C     APPLY MC CREIGHT ALGORITHM FIND UPPER BLOCK TRIANGULAR FORM
      DO 130 I=1,N
  130 IC(I)=I
      CALL EQVCL(N,M,A,IC,NSM,IA,IB,ID)
C     SCALE IRREDUCIBLE BLOCKS ON DIAGONAL
      IK=0
      DO 140 I=1,NSM
      IL=IB(I)
      IC(I)=IK+IL
      D(IK+IL)=1.
      IF(IL.LE.1) GO TO 140
      DO 135 K=1,IL
      KK=IA(IK+K)
      DO 135 L=1,IL
      LL=IA(IK+L)
  135 B(K,L)=A(KK,LL)
      CALL SCALE(B,D(IK+1),IL,M)
  140 IK=IK+IL
C     APPLY SCALE FACTORS TO ENTIRE MATRIX
      DO 142 I=1,N
  142 D(I)=1./D(I)
  143 II=1
      DO 150 I=1,NSM
      JJ=II-1+IB(I)
      DO 149 J=II,JJ
      L=IA(J)
      DO 149 K=1,N
      KK=IA(K)
      IF(K.GE.II) GO TO 145
      B(J,K)=0.
      B(J,K)=0.
      GO TO 149
  145 IF(K.NE.J) GO TO 147
      B(J,K)=A(L,KK)
      GO TO 149
  147 B(J,K)=A(L,KK)*D(K)/D(J)
  149 CONTINUE
  150 II=JJ+1
      RETURN
      END
```

```
CQNWT2
      SUBROUTINE QNWT2(X,N,NR,FUN,P,TOL,ITMAX,IPER,M,R,RMS,AJ,B,
     1 ERROR,NSIM)
C  VERSION 3.                              REVISED: JUNE 7 1976
C  PURPOSE:  SOLVE SYSTEM OF NONLINEAR ALGEBRAIC EQUATIONS.
C            0 = F(X)
C  CALL SEQUENCE:  X     - SOLUTION VECTOR UPON RETURN.  INITIAL GUESS
C                          UPON ENTRY.
C                  N     - NUMBER OF VARIABLES
C                  NR    - DIMENSION OF JACOBIAN MATRIX, AJ
C                  FUN   - NAME OF SUBROUTINE CONTAINING NONLINEAR FUNCT
C                          EVALUATION.
C                  P     - PRINT INDICATOR
C                          P = 6  CAUSES PRINT EACH ITERATION OF STATES
C                                 RATES, AND ITERATION INFORMATION.
C                          P = 7  JACOBIAN MATRIX IS ALSO PRINTED EACH
C                                 ITERATION.
C                  TOL   - CONVERGENCE TOLERANCE.
C                  ITMAX - MAXIMUM NUMBER OF ITERATIONS
C                  IPER  - INTERGER WORK ARRAY - LENGTH = N
C                  M     - INDICATOR THAT AJ CONTAINS INITIAL CALC. OF J
C                          M = 1 INDICATES THAT JACOBIAN HAS BEEN CALC.
C                  R     - VECTOR OF FUNCTION VALUES FOR CURRENT VALUE O
C                  RMS   - ROOT MEAN SQUARE OF R
C                  AJ    - JACOBIAN MATRIX
C                  B     - WORK VECTOR OF DIMENSION  N**2 + 2*N
C                  ERROR - VECTOR GIVEN RELATIVE ERROR SIZE FOR THE VARI
C                  NSIM  - TOTAL SYSTEM ORDER (INCLUDING FROZEN STATES)
C  DESIGNED BY CLAUDE GAGNON  FEB 1970
      COMMON/CINT/INT(1)/CTIME/TIME
      DIMENSION AJ(NR,1),R(1),X(1),B(N,1),IPER(1),ERROR(1)
      EXTERNAL FUN
      DATA RT,DX/0.5,.000001/
C ---       INITIALIZE
      IREC=0
      IT=0
      IP=P
      IS=0
      S1=0.
      E=.1
      NS=1
      ITMX=IABS(ITMAX)
      IF(ITMAX.LE.0) NS=2
C ---       COMPUTE INITIAL NORM OF X
      DO 5 I=1,N
      S1=S1+X(I)**2
    5 CONTINUE
      IF(S1.NE.0.)GO TO 9
      DO 7 I=1,NSIM
      IF(INT(I).EQ.0)GO TO 7
      S1=S1+ERROR(I)**2
7     CONTINUE
      S1=S1*1.E4
C ---       EVALUATE FUNCTION AT X
9     CALL EVAL2(X,NSIM,FUN,P,RMS,R)
      ICL=1
C ---       TEST FOR JACOBIAN PRINTOUT
      IF(IP.LT.6.)GO TO 10
```

```
      CALL LPRINT(2,TIME)
      IF(IP.NE.7)GO TO 10
      WRITE(6,1001)
1001  FORMAT(/50X,*INITIAL JACOBIAN*)
      DO 1005 I=1,N
      WRITE(6,1003)I,(AJ(I,J),J=1,N)
1003  FORMAT(1X,I3,(T5,10(2X,G10.4)))
1005  CONTINUE
   10 CONTINUE
C ---- 0     TEST IF INITIAL JACOBIAN IS GIVEN
      IF(M.NE.0) GO TO 33
C ----      INITIALIZE ARRAY OF X CHANGES
      DO 15 I=1,NSIM
15    B(I,N+1)=1.
C ----       CALCULATE COMPLETE JACOBIAN
20    J=0
      DO 29 J1=1,NSIM
      IF(INT(J1).EQ.0)GO TO 29
      J=J+1
      XSAVE=X(J)
      DELTA=SIGN(ERROR(J1),B(J1,N+1))
      X(J)=X(J)+DELTA
      CALL EVAL2(X,NSIM,FUN,P,RMSJAC,AJ(1,J))
      ICL=ICL+1
      X(J)=XSAVE
      DO 30 I=1,N
      AJ(I,J)=(AJ(I,J)-R(I))/DELTA
   30 CONTINUE
   29 CONTINUE
C ---- 0     TEST IF PRINT OF ITERATION INFORMATION IS REQUESTED
      IF(IP.NE.7)GO TO 33
      WRITE(6,1006)
1006  FORMAT(/50X,*RECALCULATED JACOBIAN*)
      DO 1007 I=1,N
      WRITE(6,1003)I,(AJ(I,J),J=1,N)
1007  CONTINUE
C ================  MAIN ITERATION LOOP  ===========
   33 DO 80 K=1,10
      IF(IT+K.GT.ITMX)RETURN
      IF(RMS.LE.TOL) RETURN
      STD=RMS
C ---- 0     FORM VECTOR OF RESIDUALS
      DO 35 I=1,N
      B(I,N+2)=-R(I)
      DO 34 J=1,N
      B(I,J)=AJ(I,J)
   34 CONTINUE
   35 CONTINUE
C ----       SOLVE FOR CHANGE IN X USING JACOBIAN AND RESIDUALS
      CALL LUEQS(B,B(1,N+1),B(1,N+2),IPER,N,1,N,N,1.E-16,IERROR)
C ---- 0     CALCULATE NORM OF CHANGE TO X
      S2=0.
      DO 40 I=1,N
      B(I,N+2)=R(I)
      S2=S2+B(I,N+1)**2
   40 CONTINUE
      RATIO=SQRT(S2/S1)
C ----       CALCULATE FRACTION OF STEP TO TAKE
```

```
      C=AMIN1(E/RATIO,1.)
      FRAC=C
      IH=0
      F=E
      IF(NS.EQ.1) GO TO 44
C ---         ADJUST CHANGE VECTOR AND MAKE TRIAL STEP
   44 DO 48 I=1,N
      B(I,N+1)=C*B(I,N+1)
      B(I,1)=X(I)+B(I,N+1)
   48 CONTINUE
C ---         EVALUATE FUNCTION AT TRIAL POINT
      CALL EVAL2(B,NSIM,FUN,P,RMS,R)
      ICL=ICL+1
C --- 0     IF RESIDUALS INCREASE, GO TO 58 WHERE SMALLER STEP WILL BE T
      IF(RMS.GE.STD) GO TO 58
C --- 0     ACCEPT STEP, UPDATE STEP SIZE PARAMETERS
      IREC=0
      IS=IS+1
      IF(IS.LT.2) GO TO 60
      IS=0
      TE=E+.1
      IF(TE.LE.2.) E=TE
      GO TO 60
C ---         HALVE STEP SIZE
   58 C=RT
      FRAC=FRAC*RT
      IS=0
C ---         ADJUST STEP SIZE PARAMETERS
      TE=E-0.05
      IF(TE.GE..09) E=TE
      IH=IH+1
C ---         TRY REDUCED STEP SIZE 3 TIMES
      IF(IH.LT.3) GO TO 44
C ---         IF FAILURE AFTER 3 TIMES, RECALCULATE JACOBIAN
      IF(IREC.EQ.1)GO TO 59
      IREC=1
      IT=IT+K-1
      CALL EVAL2(X,NSIM,FUN,P,RMS,R)
      GO TO 20
C ---         IF STILL NO IMPROVEMENT, TAKE STEP ANYWAY TO GET AWAY FROM
C                BAD POINT.
59    IREC=0
      FRAC=FRAC*2.
C ---         TAKE STEP
   60 D=S2*FRAC**2
      S2=1.-FRAC
C ---         UPDATE NORM OF X
      S3=0.
      DO 65 I=1,N
      X(I)=B(I,1)
      S3=S3+X(I)*X(I)
      T1=R(I)-B(I,N+2)*S2
      DO 64 J=1,N
C --- 0     MODIFY JACOBIAN VIA NEW FUNCTION INFORMATION
      AJ(I,J)=AJ(I,J)+T1*B(J,N+1)/D
   64 CONTINUE
   65 CONTINUE
      S1=AMAX1(S1,S3)
```

```
      J=IT+K
C ---       TEST FOR PRINT REQUEST
      IF(IP.LT.6)GO TO 70
      WRITE(6,200)J,RATIO,F,FRAC
  200 FORMAT(///17H ITERATION NUMBERI3/67H RATIO OF LENGTH OF FULL NEWTO
     XN STEP TO LENGTH OF INITIAL VECTOR = E18.10/24H MAXIMUM ALLOWED RA
     XTIO =E18.10,10X,31HFRACTION OF NEWTON STEP TAKEN =E18.10)
      RN=RMS*RMS
      WRITE(6,210) ICL,RN
  210 FORMAT(33H NUMBER OF FUNCTION EVALUATIONS =I4,5X,25HRESIDUAL SUM O
     XF SQUARES =E18.10/)
      CALL LPRINT(2,TIME)
      IF(IP.NE.7)GO TO 70
      WRITE(6,1008)
1008  FORMAT(/50X,*MODIFIED JACOBIAN*)
      DO 1009 I=1,N
      WRITE(6,1003)I,(AJ(I,J),J=1,N)
1009  CONTINUE
   70 IF(J.GE.ITMX) RETURN
   80 CONTINUE
C ================ END OF ITERATION LOOP ===========
C ---       TEST IF ERROR IS WITHIN TOLERANCE
      IF(RMS.LE.TOL) RETURN
      IT=IT+10
C ---       TEST IF NUMBER OF ITERATIONS EXCEEDS GIVEN MAXIMUM
      IF(IT.GE.ITMX) RETURN
      IF(RMS.LE.0.5*STD) GO TO 33
      GO TO 20
      END
```

```
CRKINIT
      SUBROUTINE RKINIT
      COMMON/CX/X(1)/CXDOT/XDOT(1)/CTIME/TIME/CORDER/NSIM,NOV,NOP
      COMMON/CWORK/A(1)/CWORKN/NN,N1,N2,N3,N4,N5,N6,N7
      COMMON/CNTRLS/ANTYPE,IPRINT,IMODE,ERROR(1)
      COMMON/CSIMUL/IPRIN,IPRATE,IOUT,NPTS,NPTMAX,INDMAX,TINC,TMAX
      COMMON/NRKVS3/IDIAG,AHMINM,AHSTRT,INTFLG,MAXN,MFAILI,MSTEP2,ISTABL
      COMMON/CDIFS/JSTART,KINIT
      IDIAG=0
      AHMINM = 0.
      AHSTRT = 0.
      INTFLG=0
      ISTABL=1
      MAXN=100
      MFAILI=10
      MSTEP2=100
      RETURN
      END
```

```
CRLBTCH
      OVERLAY(RLBTCH,11,0)
      PROGRAM RLBTCH
C   VERSION 3.                                  REVISED: APRIL 30 1976
C   PURPOSE  CONTROL AND DISPLAY CALCULATION OF ROOT LOCUS
      COMMON/CORDER/NSIM,NOV,NOP
      COMMON/CPRON/DUM1(7),RLPAR
      COMMON/CPROV/DUM2(17),START,STOP,ROOTS,RLSCL(4)
      COMMON/COVRLY/INST,LOKSS,LOKSIM
      COMMON/CWORK/A(1)
      COMMON /CWORKN/NN,N1,N2,N3,N4,N5,N6,N7
      COMMON /CXIC/XIC(1 )/CINT/INT(1 )
      COMMON /CNTRLS/ANTYPE,IPRINT,MODE,ERROR(1)
       COMMON/CSMPAR/SMPAR(10),ICSS,ICRL
      REAL NAMEX,RLPAR,SMPAR
      COMMON/CNAMEX/NAMEX(1)
      COMMON /CPLOTS/ INDPLT,INDWR,IOPT(30),PLOTID( 5),PTITLE( 8),
     +                IPOPT(10)
      DIMENSION YOPT(1)
      EQUIVALENCE (YOPT(1),IOPT(1))
      DIMENSION RLARRY(1000)
      DATA ICN/10H,IC        /,IBLNK/10H          /
      CALL CODGEN(RLPAR,ICRL,IPARAM),RETURNS(4000)
      ICX=IBLNK
      IF(ICRL.EQ.1) ICX=ICN
C    PRINT DERCAL ARRAY
      IOPT(3) = IBLNK
      IOPT(4) = IBLNK
      CALL DTTIM (IOPT(3))
      IF(INST.EQ.35)GO TO 2826
      WRITE (6,2809) PTITLE,IOPT(3),IOPT(4),
     +               (I,NAMEX(I),XIC(I),ERROR(I),INT(I),I=1,NSIM)
 2809 FORMAT(50X,30H/*/*/*/   ROOT LOCUS   /*/*/*/ //
     2 26X,8A10//54X,2A12//
     17X,5HSTATE,6X,*OPERATING   PERTURBATION   INTEGRATOR*
     2/8X,*NAME*,8X,*POINT*,9X,*SIZE*,8X,*CONTROL*/
     2 (I5,1X,A10,G14.5,G12.3,I9))
C    INITIALIZE STABILITY MATRIX CALCULATION MODE INDICATOR
      ITEST=0
      GO TO 2830
C ---------- SET MODE INDICATOR TO PREVENT RECALCULATION OF A MATRIX
2820  ITEST=3
C    INITIALIZE ROOT LOCUS DISPLAY ARRAY INDEX
2830  INDEX=1
C    INITIALIZE AUTOMATIC SCALE RANGE PARAMETERS
      XMIN=1.E36
      XMAX=-1.E36
      YMIN=1.E36
      YMAX=-1.E36
      CALL VAROUT(IPARAM,PARAMO)
      DELTA=0.
      NROOTS=ROOTS
      IF(NROOTS.GT.1) DELTA=(STOP-START)/(NROOTS-1)
C ---------  TEST TO PREVENT START FROM EQUALLING NOMINAL VALUE
      IF(START.EQ.PARAMO)START=PARAMO+DELTA
C    PARAMETER SCAN VALUE
 2850 PARAM=START
C    ROOTINGS COUNTER
```

```
      KOUNT=0
      IF(INDEX.GT.1.OR.PARAMO.EQ.0.) GO TO 2860
      KOUNT=-1
      PARAM=PARAMO
 2860 CALL VARMOD(IPARAM,PARAM)
C    SAVE PARAMETER VALUE
      RLARRY(INDEX+1)=PARAM
C    DETERMINE PLOT SYMBOL INDICATOR
      SYMB=5.
      IF(PARAM.EQ.PARAMO) SYMB=6.
      IF(PARAM.EQ.0) SYMB=7.
C    STORE SYMBOL INDICATOR IN DISPLAY ARRAY
      RLARRY(INDEX+2)=SYMB
C    SAVE INDEX FOR STARTING POINT OF CURRENT SET OF ROOTS
      INDEXS=INDEX
C    PRINT PARAMETER INDICATOR AND ROOTS
      WRITE(6,2880) RLPAR,ICX,PARAM
 2880 FORMAT(//22X,A8,A4,,3H = ,G12.6)
      IF(PARAM.EQ.PARAMO) WRITE(6,2882)
 2882 FORMAT(*+    NOMINAL VALUE*)
      CALL ROTCAL(NSIM,        INDEX,ITEST,RLARRY,XMIN,XMAX,YMIN,YMAX,
     1 A,A(NN),A(N1),A(N2),A(N3),A(N4),A(N5),A(N2),A(N3),A(N4),A(N5),
     2 A(N1))
C    CAUSE ITEST TO TAKE ON VALUES 0,2,3,3,.......
      IF(ITEST.EQ.2) ITEST=3
      IF(ITEST.EQ.0) ITEST=2
 2975 IF(KOUNT.LE.-1) GO TO 2850
C    ADVANCE ROOTING COUNTER
      KOUNT=KOUNT+1
      IF(KOUNT.GE.NROOTS) GO TO 3020
      PARAM=PARAM+DELTA
      GO TO 2860
C    RESTORE PARAMETER TO NOMINAL VALUE
 3020 CALL VARMOD(IPARAM,PARAMO)
C
C      SET PLOT PARAMETERS.
C
      IF ( INDPLT .EQ. 0 ) GO TO 6000
      IOPT(1) = 3
      IOPT(2) = IOPT(2) + 1
      YOPT(5) = RLPAR
      DO 150 I=6,11
  150 YOPT(I) = 0.0
      IF ( IPOPT(1) .EQ. 0 ) GO TO 200
      YOPT( 6) = RLSCL(1)
      YOPT( 7) = RLSCL(2)
      YOPT( 9) = RLSCL(3)
      YOPT(10) = RLSCL(4)
  200 CONTINUE
      NPTS = INDEX-1
      IOPT(12) = NPTS
      WRITE (30) IOPT,PLOTID,PTITLE
      WRITE (30) (RLARRY(I),I=1,NPTS)
      INDWR = 1
      WRITE(6,4011)
4011  FORMAT(/////)
      GO TO 6000
 4000 WRITE(6,4001) RLPAR,ICX
```

```
 4001 FORMAT(//10X,31H*** WARNING ***  CAN*T IDENTIFY,1X,A8,A4,1X,
     1 31HAS A VALID ROOT LOCUS PARAMETER//)
      WRITE(6,4011)
6000  CONTINUE
      END
```

```
CROTCAL
      SUBROUTINE ROTCAL(NSIM,INDEX,ITEST,RLARRY,XMIN,XMAX,YMIN,YMAX,
     1 A,RATIO,DWORK,IA,IB,IC,ID,EVR,EVI,WN,DAMPR,XDOTO)
C  VERSION 2.                          REVISED: OCT 5 1976
C  PURPOSE--CALCULATE STABILITY MATRIX AND IT*S EIGENVALUES AND RETURN
C           ALL EIGENVALUES WITH IMAGINARY PARTS >=0.
C  CALL SEQUENCE
C   NSIM   = TOTAL NONLINEAR SYSTEM ORDER.
C   INDEX  = STARTING LOCATION IN ARRAY RLARRY FOR RETURNED DATA.
C   ITEST  = INDICATOR FOR MODE OF OPERATION FOR STABMX ROUTINE.
C            (THIS MUST BE 0,2, OR 3)
C   RLARRY = ARRAY OF ROOT LOCUS DATA.
C            FIRST WORD = NO. OF POLES
C            SECOND WORD = PARAMETER VALUE
C            THIRD WORD = SYMBOL DESIGNATOR
C            NEXT N WORDS = REAL PARTS OF POLES (N = NO. OF POLES)
C            NEXT N WORDS = IMAGINARY PARTS OF POLES
C            THIS PATTERN IS REPEATED FOR EACH SET OF POLES
C  LIMITATIONS
C   1. THE DIMENSION OF THE LOCAL ARRAY IJK LIMITS THE MAXIMUM NUMBER
C      OF ELEMENTS OF A THAT CAN BE EFFECTED BY A ROOT LOCUS PARAMETER
C      TO 400.  IF MORE ELEMENTS ARE EFFECTED, A SLOWER METHOD IS USED
C      TO CALCULATE A.
C   2. ICOUNT + IJK ARE LOCAL VARIABLES THAT ARE REQ*D ON SUBSEQUENT
C      CALLS, THUS PRECLUDING OVERLAYING THIS ROUTINE DURING ROOT LOCUS
C      CALCULATIONS.
      DIMENSION RLARRY(1),IJK(400)
      DIMENSION EVR(1),IA(1),IB(1),IC(1),ID(1),XDOTO(1),DWORK(1)
      DIMENSION EVI(1),A(1),RATIO(1)
      DIMENSION WN(1),DAMPR(1)
      DATA IPOM/4H+-+-/,IBLNK/4H    /
C   CALCULATE STABILITY MATRIX
      CALL STABMX(NSIM,XDOTO,ICOUNT,IJK,A,NLIN,ITEST)
C   CALCULATE EIGENVALUES
      CALL EGVL3(A,RATIO,EVR,EVI,IA,IB,IC,ID,DWORK,1.E-14,NLIN,NLIN)
      CALL NATFRQ(EVR,EVI,WN,DAMPR,NLIN,NPOLES)
C   STORE NO. OF POLES
      RLARRY(INDEX)=NPOLES
C   INITIALIZE REAL + IMAGINARY PART INDICES
      IR=INDEX+3
      II=IR+NPOLES
      WRITE(6,2867) NLIN
 2867 FORMAT(/   28X,I3,2X,*EIGENVALUES*/13X,*REAL*,9X,*IMAGINARY*,
     1 6X,*NATURAL FREQ.*,5X,*DAMPING RATIO*)
C   STORE REAL + IMAGINARY PARTS OF POLES
      DO 200 I=1,NPOLES
      DUM=EVR(I)
      RLARRY(IR)=DUM
      IR=IR+1
      IF(DUM.LT.XMIN) XMIN=DUM
      IF(DUM.GT.XMAX) XMAX=DUM
      DUM=EVI(I)
      RLARRY(II)=DUM
      II=II+1
      IF(DUM.LT.YMIN) YMIN=DUM
      IF(DUM.GT.YMAX) YMAX=DUM
      J=IBLNK
      IF(EVI(I).GT.0.) J=IPOM
```

```
2868 WRITE(6,2869)I,EVR(I),J,EVI(I),WN(1),DAMPR(I)
2869 FORMAT(3X,I3,3X,G12.6,2X,A2,G12.6,4X,2G16.6)
 200 CONTINUE
     INDEX=II
     RETURN
     END
```

```
CSCALE
      SUBROUTINE SCALE(A,T,N,M)
C     SUBROUTINE TO SCALE AN  IRREDUCIBLE MATRIX A.
C     T IS AN M DIMENSIONAL VECTOR CONTAINING THE DIAGONAL
C     COMPONENTS OF THE TRANSFORMATION MATRIX
C     N IS THE ORDER OF A
C     M IS THE ROW DIMENSION OF A
C
C     THIS PROGRAM WAS DESIGNED AND CODED BY A. FREDERICK FATH OF
C     BOEING COMPUTER SERVICES, SEATTLE, WASHINGTON.  THIS VERSION
C     WAS COMPLETED DURING APRIL 1975.
C
      DIMENSION A(M,1),T(1)
      N1=N-1
      IT=0
      DO 5 I=1,N
 5    T(I)=1.
 6    IT=IT+1
      IF(IT.GE.M) GO TO 40
      TMI=0.
C     FORM ROW AND COLUMN E-NORMS
      DO 30 K=1,N
      SR=0.
      SS=0.
      DO 10 J=1,N
      IF(J.EQ.K) GO TO 10
      SS=SS+A(J,K)**2
      SR=SR+A(K,J)**2
 10   CONTINUE
      SS=SQRT(SS/SR)
C     DETERMINE COMPONENT MULTIPLIER
      TM=SQRT(SS)
      IF(ABS(TM-1.).GT.TMI) TMI=ABS(TM-1.)
C     ADJUST A MATRIX AND TRANSFORMATION T
      IF(K-N) 12,20,20
 12   T(K)=T(K)*TM
      DO 15 J=1,N
      IF(J.EQ.K) GO TO 15
      A(K,J)=A(K,J)*TM
      A(J,K)=A(J,K)/TM
 15   CONTINUE
      GO TO 30
 20   DO 22 J=1,N1
 22   T(J)=T(J)/TM
      DO 25 I=1,N1
      A(I,N)=A(I,N)/TM
 25   A(N,I)=A(N,I)*TM
 30   CONTINUE
      IF(TMI.GT..1) GO TO 6
   40 CONTINUE
      RETURN
      END
```

```
CSETIN
      SUBROUTINE SETIN(I,VAR)
C   PURPOSE:  TO MODIFY THE CURRENT VALUE OF A STATE VARIABLE,PARAMETER,
C             ETC. AND TO EXECUTE THE MODEL TO OBSERVE THE RESULTS OF
C             THE MODIFICATION.
C   CALL SEQUENCE:  I   = IDENTIFICATION CODE.
C                   VAR = NEW NUMERIC VALUE OF QUANTITY IDENTIFIED BY COD
      COMMON/CX/X(1)/CXDOT/XDOT(1)/CV/V(1)/CP/P(1)/CXIC/XIC(1)
      COMMON/CTIME/TIME
C   TEST FOR TIME
      IF(I.NE.0) GO TO 10
      TIME=VAR
5     CALL EQMO(0.,0.,0)
      RETURN
C   TEST FOR STATES
10    IF(I.LT.1.OR.I.GT.1000000) GO TO 20
      X(I)=VAR
      GO TO 5
C   TEST FOR VARIABLES
20    IF(I.LE.3000000.OR.I.GT.4000000) GO TO 30
      J=I-3000000
      V(J)=VAR
      CALL VARSET(0.,0.,J)
      RETURN
C   TEST FOR RATES
30    IF(I.LE.1000000.OR.I.GT.2000000) GO TO 40
      J=I-1000000
      XDOT(J)=VAR
      CALL RATSET(0.,0.,J)
      RETURN
C   TEST FOR PARAMETERS
40    IF(I.LE.4000000.OR.I.GT.5000000) RETURN
      P(I-4000000)=VAR
      GO TO 5
      END
```

```
CSHELLX
      SUBROUTINE SHELLX(DARRAY,KEY,N)
C  PURPOSE:  REORDER ELEMENTS OF SINGLE DIMENSION ARRAY
C            BASED ON THE INDEX ARRAY KEY.
C  CALL SEQUENCE:  DARRAY - ARRAY TO BE REORDERED
C                  KEY    - INDEX ARRAY
C                  N      - NUMBER OF ELEMENTS IN ARRAY
      DIMENSION DARRAY(1),KEY(1)
      IFIRST=1
   10 DO 20 I=IFIRST,N
      IF(KEY(I))20,20,40
   20 CONTINUE
      DO 30 I=1,N
   30 KEY(I)=-KEY(I)
      RETURN
   40 IFIRST=I
      TEMP=DARRAY(I)
      GO TO 60
   50 DARRAY(I)=DARRAY(IK)
      I=IK
   60 IK=KEY(I)
      KEY(I)=-IK
      IF(IK-IFIRST)50,70,50
   70 DARRAY(I)=TEMP
      GO TO 10
      END
```

```
CSIBTCH
      OVERLAY(SIBTCH,4,0)
      PROGRAM SIBTCH
C  VERSION 3.1                              REVISED: OCT 7 1976
      COMMON/CORDER/NSIM,NOV,NOP
      COMMON/CPROV/DUM1(8),AMODE,DUM2(15)
      COMMON /CX/X(1)/CXDOT/XDOT(1)/CINT/INT(1)/CXIC/XIC(1)
      COMMON /CNTRLS/ANTYPE,IPRINT,IMODE,ERROR(1)
      COMMON /CSIMUL/IPRIN,IPRATE,IOUT,NPTS,NPTMAX,INDMAX,TINC,TMAX
     1 ,INDEX,IPLOT,IDENT(4)
      COMMON/CPRINT/PRTNAM(10),LPRT(10)/CDIFS/JSTART,KINIT,TP
      REAL IDENT
      COMMON/CTIME/TIME/ERMESS/IFATAL,IERR
      COMMON /CWORK/ DSPLY(1)
      COMMON /CSCALE/ SCALE(5,4,6),NVAR( 5,2,6),NPLTS(6)
      COMMON /CPLOTS/ INDPLT,INDWR,IOPT(30),PLOTID( 5),PTITLE( 8),
     +                IPOPT(10)
      DIMENSION IVAR(5,2,6),IVRCOD(31)
      DIMENSION VRCOD(31)
      DATA IBLK /10H          /
      IMODE=AMODE
      IPLOT=1
      ICOUNT=0
      DO 5 I=1,31
    5 VRCOD(I) = 0.0
      IOPT(3) = IBLK
      IOPT(4) = IBLK
      CALL DTTIM (IOPT(3))
      IOPT(2)=IOPT(2)+1
      WRITE(6,2708)IPRATE,IOUT,IMODE,TINC,TMAX,PTITLE,(IOPT(I),I=2,4)
 2708 FORMAT(45X,41H/*/*/*/*/   SIMULATION ANALYSIS   /*/*/*/ //20X,
     1 11HPRINT RATE=,I3,3X,13HDISPLAY RATE=,I3,3X, 5HMODE=,
     2 I3,3X, 5HTINC=, G12.5,3X, 5HTMAX=, G12.5//26X,8A10//
     3 15X,*CASE NO.*,I4,27X,2A12/)
      IPOUT=IOUT*IPRATE
      IPRCNT=0
      INDEX=1
      ISET=0
      IF ( INDPLT .EQ. 0 ) GO TO 67
C
C     FIND CODE NUMBERS FOR THIS SIMULATION.
C
C         NVAR - PARAMETER NAMES FOR EACH PLOT
C         IVAR - POINTERS INTO IVRCOD FOR EACH PARAMETER
C       IVRCOD - UNIQUE CODE NUMBERS USED IN THIS SIMULATION
C
      NCODES = 1
      NDISP = 0
      IVRCOD(1) = 0
      DO 65 J=1,6
      IMAX = NPLTS(J)
      IF ( IMAX .EQ. 0 ) GO TO 65
      NDISP = J
      DO 60 I=1,IMAX
      CALL CODGEN (NVAR( I,1,J),0,IV1), RETURNS(10)
   10 CALL CODGEN (NVAR( I,2,J),0,IV2), RETURNS(20)
   20 CONTINUE
      DO 30 K=1,NCODES
```

```
      IF ( IVRCOD(K) .NE. IV1 ) GO TO 30
      IVAR(I,1,J) = K
      GO TO 40
   30 CONTINUE
      NCODES = NCODES + 1
      IVRCOD(NCODES) = IV1
      IVAR(I,1,J) = NCODES
   40 CONTINUE
      DO 50 K=1,NCODES
      IF ( IVRCOD(K) .NE. IV2 ) GO TO 50
      IVAR(I,2,J) = K
      GO TO 60
   50 CONTINUE
      NCODES = NCODES + 1
      IVRCOD(NCODES) = IV2
      IVAR(I,2,J) = NCODES
   60 CONTINUE
   65 CONTINUE
   67 CONTINUE
C
C     INITIALIZE FOR SIMULATION
C
      DO 70 I=1,NSIM
      X(I)=XIC(I)
   70 XDOT(I)=0.
      JSTART=0
      KINIT=0
C ---------- TURN ON ERROR MESSAGES IN MODEL
      IERR=1
      CALL EQMO(TIME,TMAX,ISET)
C ---------- TURN OFF ERROR MESSAGES IN MODEL
      IERR=0
      IF(IPRIN.GT.0)CALL LPRINT(IPRIN,TIME)
      IF ( INDPLT .EQ. 0 ) GO TO 77
      DO 75 K=1,NCODES
      CALL VAROUT (IVRCOD(K),VRCOD(K))
   75 CONTINUE
      WRITE (25) VRCOD
   77 CONTINUE
C
C     INCREMENT COUNTERS AND SAVE PARAMETER VALUES IF REQUIRED.
C
   80 CALL STEP1(TIME,TINC)
      ICOUNT=ICOUNT+1
      IPRCNT=IPRCNT+1
      IF(ICOUNT.LT.IOUT) GO TO 130
      ICOUNT=0
      IF ( INDPLT .EQ. 0 ) GO TO 105
      INDEX=INDEX+1
      DO 100 K=1,NCODES
      CALL VAROUT (IVRCOD(K),VRCOD(K))
  100 CONTINUE
      WRITE (25) VRCOD
  105 CONTINUE
      IF(IPRCNT.LT.IPOUT) GO TO 130
      IPRCNT=0
      IF ( IPRIN .GT. 0 ) CALL LPRINT (IPRIN,TIME)
      GO TO 130
```

```
  110 CONTINUE
      WRITE (6,120)
  120 FORMAT (///1H ,10(1H*),7HWARNING, 10(1H*),66H THE NUMBER OF DATA P
     +OINTS EXCEEDS AVAILABLE STORAGE FOR ONE RUN. ,20(1H*)//
     +28X,40H THE DATA TO THIS POINT WILL BE PLOTTED.///)
      INDEX = INDEX - 1
      GO TO 140
  130 CONTINUE
      IF(TIME.LT.TMAX -.00001) GO TO 80
  140 CONTINUE
      WRITE(6,2941)
2941  FORMAT(/////)
C
C     WRITE PLOT DATA.
C
      IF ( INDPLT .EQ. 0 ) GO TO 200
      IOPT(1) = 2
      IOPT(5) = NDISP
      DO 150 I=1,NDISP
      IOPT(5 I) = NPLTS(I)
  150 CONTINUE
      IOPT(12) = INDEX
      IOPT(13) = NCODES
      IOPT(14) = IPOPT(2)
      IOPT(15) = NWORK
      WRITE (30) IOPT,PLOTID,PTITLE
      WRITE (30) SCALE,NVAR,IVAR
      REWIND 25
      DO 180 I=1,INDEX
      READ (25) VRCOD
      WRITE (30) VRCOD
  180 CONTINUE
      REWIND 25
      INDWR = 1
  200 CONTINUE
      END
```

```
CSLVEQ
      SUBROUTINE SLVEQ(A,B,C,IA,NA,NB,MA,MB,MC,FPZ,IERROR)
C     SOLVES A*B=C WHEN LUEQS HAS ALREADY BEEN CALLED FOR THE GIVEN A
C     A AND IA MUST BE THE SAME AS RETURNED FROM THE PREVIOUS CALL.
      DIMENSION A(MA,1),B(MA,1),C(MC,1),IA(1)
      IERROR=0
      ERR=FPZ*50.
      IF(NB.EQ.0) RETURN
      NA1=NA-1
C     PERMUTE ROWS OF C
      DO 70 I=1,NA
      K=IA(I)
      DO 70 J=1,NB
 70   B(I,J)=C(K,J)
      IF(NA.LE.1) GO TO 76
C     SOLVE FORWARD SUBSTITUTION
      DO 75 J=1,NB
      DO 75 I=2,NA
      SUM=B(I,J)
      SB=ABS(SUM)
      IK=I-1
      DO 74 K=1,IK
      SA=A(I,K)*B(K,J)
      IF(ABS(SA).GT.SB) SB=ABS(SA)
 74   SUM=SUM-SA
      IF(ABS(SUM).LT.ERR*SB) SUM=0.
 75   B(I,J)=SUM
C     SOLVE BACK SUBSTITUTION
 76   DO 77 J=1,NB
      B(NA,J)=B(NA,J)*A(NA,NA)
 77   CONTINUE
      IF(NA.LE.1) RETURN
      DO 64 I=1,NA1
      J=NA-I
      J1=J+1
      DO 64 K=1,NB
      SUM=B(J,K)
      SB=ABS(SUM)
      DO 62 L=J1,NA
      SA=A(J,L)*B(L,K)
      IF(ABS(SA).GT.SB) SB=ABS(SA)
 62   SUM=SUM-SA
      IF(ABS(SUM).LT.ERR*SB) SUM=0.
 64   B(J,K)=SUM*A(J,J)
      RETURN
      END
```

```
CSMBTCH
      OVERLAY(SMBTCH,6,0)
      PROGRAM SMBTCH
C  VERSION 2.                          REVISED: DEC 8 1975
C  PURPOSE:  CONTROL STABILITY MARGIN CALCULATION
      COMMON/CORDER/NSIM,NOV,NOP
      COMMON/CSMPAR/PARAM(10),ICIND(2)
      COMMON /CXIC/XIC(1 )/CINT/INT(1 )
      COMMON /CNTRLS/ANTYPE,IPRINT,MODE,ERROR(1)
      COMMON/CNAMEX/NAMEX(1)
      COMMON /CWORK/A(1)
      COMMON /CWORKN/NN,N1,N2,N3,N4,N5,N6,N7
      DIMENSION LICH(10),GMDSPY(50)
      IPOLE=0
      K1=0
C ==========  GENERATE CODES CORRESPONDING TO SM PARAMETERS
      DO 2820 I=1,10
      CALL CODGEN(PARAM(I),0,LICH(I)),RETURNS(2810)
      IF(LICH(I).LT.1) GO TO 2830
      GO TO 2815
 2810 WRITE(6,2811)PARAM(I)
 2811 FORMAT(///10X,31H*** WARNING ***  CAN*T IDENTIFY,1X,A8,1X,
     1 38HAS A VALID STABILITY MARGIN PARAMETER//)
 2815 K1=K1+1
 2820 CONTINUE
 2830 CONTINUE
      IF(K1.LE.0) GO TO 2930
      WRITE(6,2918) (PARAM(I),I=1,K1)
 2918 FORMAT(1H1 /50X,7H/*/*/*/,3X,*STABILITY MARGIN ANALYSIS*,3X,
     17H/*/*/*///10X,14HFOR PARAMETERS,3X,10A11)
      WRITE(6,2809)(I,NAMEX(I),XIC(I),ERROR(I),INT(I),I=1,NSIM)
 2809 FORMAT(/
     17X,5HSTATE,6X,*OPERATING   PERTURBATION   INTEGRATOR*
     2/8X,*NAME*,8X,*POINT*,9X,*SIZE*,8X,*CONTROL*/
     2 (I5,1X,A10,G14.5,G12.3,I9))
C ==========  CALCULATE STABILITY MARGINS
 2920 CALL GANMAR(NSIM,LICH,PARAM,K1,IPOLE,GMDSPY,A,A(NN),A(N1),
     1 A(N2),A(N3),A(N4),A(N5),A(N6),A(N2),A(N3),A(N1),A(N4)),
     1 RETURNS(3000)
      WRITE(6,2921)
 2921 FORMAT(///,28X,*SUMMARY OF STABILITY MARGIN STUDY*/
     1 92H PARAMETER  NOMINAL VALUE     LOWER MARGIN     LOWER FREQ.
     2  UPPER MARGIN     UPPER FREQ. )
      WRITE(6,2923)(PARAM(I),(GMDSPY(I+(J-1)*K1),J=1,5),
     1 I=I,K1)
 2923 FORMAT(2X,A8,4X,G13.4,4G17.4)
 2930 WRITE(6,2931)
 2931 FORMAT(/////)
      GO TO 6000
 3000 GO TO 2920
 6000 CONTINUE
      END
```

```
CSSBTC
      OVERLAY(SSBTCH,10,0)
      PROGRAM SSBTC
C  PURPOSE:  PROVIDE OVERLAY INTERFACE TO PASS WORK STORAGE
C            AREAS TO STEADY STATE ROUTINE  SSBTCH
C  DESIGNED BY: J.D. BURROUGHS                FEB 1974
      COMMON/CWORK/A(1)
      COMMON/COVRLY/INST,LOKSS,LOKSIM
      COMMON/CORDER/NSIM,NOV,NOP
      COMMON/CPRON/DUM1(6),INDEP,DUM2
      COMMON/CPROV/DUM3(5),PRINT,DUM4(7),XSTART,XSTOP,POINTS,LIMIT,
     1 DUM5(3)
      COMMON /CWORKN/NN,N1,N2,N3,N4,N5,N6,N7
      REAL INDEP
      CALL SSBTCH(INDEP,XSTART,XSTOP,POINTS,LIMIT,NSIM,PRINT,
     1 A,A(NN),A,A(N1),A(N2),A(N3),A(N4),A(N5),A(N2),A(N3),A(N1),A(N4),
     2 A(N5),A(N6)),RETURNS(100)
      LOKSS=1
      GO TO 6000
100   LOKSS=0
6000  CONTINUE
      END
```

```
CSSBTCH
      SUBROUTINE SSBTCH(INDEP,XSTART,XSTOP,POINTS,LIMIT,NSIM,PRINT,
     1 A,RATIO,DSPLY,DWORK,IA,IB,IC,ID,EVR,EVI,XDOTO,WN,DAMPR,XICSAV),
     2 RETURNS(R1)
C  VERSION 3.2                          REVISED: OCT 8 1976
C  PURPOSE:  CONTROL CALCULATION AND DATA DISPLAY FOR STEADY
C            STATE ANALYSIS.
C  CALL SEQUENCE:  INDEP  - SS PARAMETER  (HOLLERITH NAME)
C                  XSTART - SS START
C                  XSTOP  - SS STOP
C                  POINTS - SS POINTS
C                  LIMIT  - SS ITERATIONS
C                  NSIM   - MODEL ORDER
C                  PRINT  - PRINT CONTROL
C                NAME          DIMENSION            LOCATION
C                  A       - NSIM X NSIM          /CWORK/A(1)
C                  RATIO   - NSIM X NSIM          /CWORK/A(NN)
C                  DSPLY   - NSIM X NSIM          /CWORK/A(1)  (NOT USED)
C                  DWORK   - NSIM X 1             /CWORK/A(N1)
C                  IA      - NSIM X 1              /CWORK/A(N1)
C                  IB      - NSIM X 1              /CWORK/A(N3)
C                  IC      - NSIM X 1              /CWORK/A(N4)
C                  ID      - NSIM X 1              /CWORK/A(N5)
C                  EVR     - NSIM X 1               /CWORK/A(N2)
C                  EVI     - NSIM X 1              /CWORK/A(N3)
C                  XDOTO   - NSIM X 1              /CWORK/A(N1)
C                  WN      - NSIM X 1              /CWORK/A(N4)
C                  DAMPR   - NSIM X 1              /CWORK/A(N5)
C                  XICSAV  - NSIM X 1              /CWORK/A(N6)
C            RETURN R1  - - RETURN TAKEN IF SYSTEM IS UNSTABLE
C  DESIGNED BY: J.D. BURROUGHS                    FEB 1974
C  VERSION 2.                          REVISED: SEPT 10 1975
      EXTERNAL EQMO
      COMMON /CX/X(1)/CXDOT/XDOT(1)/CINT/INT(1)/CXIC/XIC(1)
      COMMON /CNTRLS/ANTYPE,IPRINT,IMODE,ERROR(1)
      COMMON/CNAMEX/NAMEX(1)/CNAMER/NAMER(1)                              02-1
      COMMON /CSIMUL/IPRIN,IPRATE,IOUT,NPTS,NPTMAX,INDMAX,TINC,TMAX
     1 ,INDEX,IPLOT,IDENT(4)
      COMMON/CTIME/TIME
      COMMON/CPRINT/PRTNAM(10),LPRT(10)
      COMMON/CSMPAR/SMPAR(10),ICSS,ICRL
      COMMON /CSCALE/ SCALE(5,4,6),NVAR( 5,2,6),NPLTS(6)
      COMMON /CPLOTS/ INDPLT,INDWR,IOPT(30),PLOTID( 5),PTITLE( 8),
     +                IPOPT(10)
      DIMENSION DSPLY(1)
      DIMENSION IVAR(5,2,6),IVRCOD(31)
      DIMENSION NVRDMY( 5,2,6)
      REAL MVAR,NVRDMY
      DIMENSION SSAVE(31)
      DIMENSION DWORK(1),IA(1),IB(1),IC(1),A(1),RATIO(1),XICSAV(1),
     1 ID(1),EVR(1),EVI(1),XDOTO(1),WN(1),DAMPR(1)
      REAL    NAMEX,NAMER  ,INDEP,SMPAR
      REAL LIMIT
      DATA ICN/4H,IC /,IBLNK/10H          /,IPOM/10H+-+-+-+-+-/
      REWIND 25
C =========  GENERATE CODES FOR PLOTTED QUANTITIES
C ==========  DETERMINE CODE OF STEADY STATE PARAMETER
      CALL CODGEN(INDEP,ICSS,IND),RETURNS(4100)
```

```
      INDEX=0
C ==========  IF NO STEADY STATE PARAMETER, SKIP PLOTTING
      IF ( IND.LT.0 ) GO TO 70
C     FIND CODE NUMBERS FOR THIS STEADY STATE
C
C         NVAR - PARAMETER NAMES FOR EACH PLOT
C         IVAR - POINTERS INTO IVRCOD FOR EACH PARAMETER
C       IVRCOD - UNIQUE CODE NUMBERS USED IN THIS SIMULATION
C
      NCODES = 1
      NDISP = 0
      IVRCOD(1) = IND
      DO 65 J=1,6
      IMAX = NPLTS(J)
      IF ( IMAX .EQ. 0 ) GO TO 65
      NDISP = J
      DO 60 I=1,IMAX
      CALL CODGEN (NVAR( I,1,J),0,IV1), RETURNS(20)
   20 CONTINUE
      DO 30 K=1,NCODES
      IF ( IVRCOD(K) .NE. IV1 ) GO TO 30
      IVAR(I,1,J) = K
      GO TO 40
   30 CONTINUE
      NCODES = NCODES + 1
      IVRCOD(NCODES) = IV1
      IVAR(I,1,J) = NCODES
   40 CONTINUE
      IVAR(I,2,J) = 1
   60 CONTINUE
   65 CONTINUE
   70 CONTINUE
C
C =========  GENERATE LINEAR STABILITY MATRIX
C
      CALL STABMX(NSIM,XDOTO,ICOUNT,RATIO,A,NLIN,0)
C ==========  CONVERT PRINT CONTROL TO INTEGER
 2160 IPRINT= PRINT
C ========  SET OUTPUT AND LIMIT CONTROLS
      IPRIN  = IPRINT
      ITLIM=IFIX(LIMIT)
      IF(POINTS.LT.2.) POINTS=2.
      IPOINT=IFIX(POINTS)
      IF(IPOINT.GT.INDMAX) IPOINT=INDMAX
C =========  SAVE INITIAL OPERATING POINT
      CALL XFR(XIC,XICSAV,NSIM)
      IF(IND.GE.0) GO TO 2168
C =========  PRINT TITLE IF NO SS PARAMETER WAS GIVEN
      WRITE(6,2166)ITLIM
 2166 FORMAT(1H1 /30X,7H/*/*/*/,3X,*STEADY STATE ANALYSIS*,3X,7H/*/*/*/
     1//30X,*A MAXIMUM OF *,I4,* ITERATIONS CAN BE USED*/)
      GO TO 3100
 2168 ICX=IBLNK
      IOPT(2)=IOPT(2)+1
C =========  TEST IF SS PARAMETER IS STATE
      IF(ICSS.EQ.1) ICX=ICN
      IOPT(3) = IBLNK
      IOPT(4) = IBLNK
```

```
      CALL DTTIM (IOPT(3))
C ========  PRINT TITLE WITH SS PARAMETER
      WRITE(6,2170)INDEP,ICX,XSTART,XSTOP,ITLIM,PTITLE,IOPT(2),IOPT(3),
     1 IOPT(4)
 2170 FORMAT(45X,41H/*/*/*/   STEADY STATE ANALYSIS   /*/*/*/
     1 //30X,8HVARIED  ,A8,A4,5HFROM ,G10.4,4H TO ,G10.4,
     2 4X,*A MAXIMUM OF *,I4,
     3 * ITERATIONS CAN BE USED PER ANALYSIS*//26X,8A10//
     4 15X,*CASE NO.*,I4,27X,2A12)
 1150 DELTA=(XSTOP-XSTART)/(IPOINT-1)
      INDEX=1
      IPLOT=1
      PARAM=XSTART
      CALL VAROUT(IND,PARSAV)
 3000 CALL VARMOD(IND,PARAM)
      DO 3020 I=1,NSIM
 3020 X(I)=XIC(I)
      WRITE(6,3050) INDEP,ICX,PARAM
 3050 FORMAT(4H0   ,A8 ,A4,3H = ,G12.6)
C ========  SET STEADY STATE CALCULATION FLAG = 1 (SUCCESS)
3100  ISS=1
3120  P=IPRINT
      ITMAX=-IABS(ITLIM)
C ==========  CALCULATE INITIAL JACOBIAN OF SYSTEM
      IF(INDEX.GT.1)CALL STABMX(NSIM,XDOTO,ICOUNT,RATIO,A,NLIN,0)
C ========  TRANSFER X TO DAMPR AND COMPRESS OUT FROZEN STATES
      J=0
      DO 3130 I=1,NSIM
      IF(INT(I).EQ.0)GO TO 3130
      J=J+1
      DAMPR(J)=X(I)
3130  CONTINUE
C =========  CALL NONLINEAR SIMULTANEOUS EQUATION SOLVER
      CALL QNWT2(DAMPR,NLIN,NLIN,EQMO,P,.0001,ITMAX,IC,1,IB,RMS,A,
     1 RATIO,ERROR,NSIM)
C ==========  TRANSFER STEADY STATE TO STATE VECTOR X AND UNCOMPRESS
      J=0
      DO 3145 I=1,NSIM
      IF(INT(I).EQ.0)GO TO 3140
      J=J+1
      X(I)=DAMPR(J)
      GO TO 3145
3140  X(I)=XICSAV(I)
3145  CONTINUE
C =======  TEST FOR SUCCESSFUL STEADY STATE CALCULATION
      IF(P.NE.-1)GO TO 3170
C ========  SET STEADY STATE FLAG = 0  (FAILURE)
3150  ISS=0
      WRITE(6,3160)
3160  FORMAT(/5X,15H*** WARNING ***,5X,*CONVERGENCE CRITERIA NOT SATISFI
     1ED.  SOLUTION MAY BE INVALID*/)
      GO TO 4000
C ========  TEST IF SS PARAMETER WAS SPECIFIED
3170  IF(IND.LT.0) GO TO 4000
      IF(ITLIM.EQ.0) INDEX=IPOINT
C ==========  TEST TO ASSURE THAT RATES ARE PRINTED
      IF(IPRINT.EQ.5)CALL LPRINT(2,TIME)
      CALL LPRINT(IPRINT,TIME)
```

```
C
C ========  SAVE PLOT DATA
      IF ( INDEX .GT. INDMAX ) GO TO 110
      IF ( INDPLT .EQ. 0 ) GO TO 130
      DO 100 K=1,NCODES
      CALL VAROUT (IVRCOD(K),SSAVE(K))
  100 CONTINUE
      WRITE (25) SSAVE
      GO TO 130
  110 CONTINUE
      WRITE (6,120)
  120 FORMAT (///1H ,10(1H*),7HWARNING, 10(1H*),66H THE NUMBER OF DATA P
     +OINTS EXCEEDS AVAILABLE STORAGE FOR ONE RUN. ,20(1H*)//
     +28X,40H THE DATA TO THIS POINT WILL BE PLOTTED.///)
      GO TO 2783
  130 CONTINUE
      IF(INDEX.LT.IPOINT)  GO TO 2706
C ========  RESTORE SS PARAMETER TO ORIGINAL VALUE
 2783 CALL VARMOD(IND,PARSAV)
C ==========  RESTORE OPERATING POINT TO ORIGINAL VALUE
      CALL XFR(XICSAV,XIC,NSIM)
      WRITE(6,2871)
C
C     WRITE PLOT DATA.
C
      IF(IND.LT.0.OR.INDPLT.EQ.0)GO TO 200
C =========  TRANSFER PLOT DATA TO PLOT TAPE -- TAPE30
      IOPT(1) = 5
      IOPT(5) = NDISP
      DO 150 I=1,NDISP
      IOPT(5+I) = NPLTS(I)
  150 CONTINUE
      IOPT(12) = INDEX
      IOPT(13) = NCODES
      IOPT(14) = IPOPT(3)
      DO 155 K=1,NDISP
      DO 153 I=1,5
      NVRDMY(I,1,K) = NVAR(I,1,K)
  153 CONTINUE
      DO 155 I=1,5
      NVRDMY(I,2,K) = INDEP
  155 CONTINUE
      WRITE (30) IOPT,PLOTID,PTITLE
      WRITE (30) SCALE,NVRDMY,IVAR
      REWIND 25
      DO 180 I=1,INDEX
      READ (25) SSAVE
      WRITE (30) SSAVE
  180 CONTINUE
      REWIND 25
      INDWR = 1
  200 CONTINUE
      IF(ISS.EQ.0)RETURN R1
C
      RETURN
4000  IF(IPRINT.EQ.5)CALL LPRINT(2,TIME)
      CALL LPRINT(IPRINT,TIME)
C ==========  SET XIC = STEADY STATE OPERATING POINT
```

```
      CALL XFR(X,XIC,NSIM)
C ========  GENERATE LINEAR STABILITY MATRIX
      CALL STABMX(NSIM,XDOTO,ICOUNT,RATIO,A,NLIN,0)
C =========  CALCULATE EIGENVALUES
2862  CALL EGVL3(A,RATIO,EVR,EVI,IA,IB,IC,ID,DWORK,1.E-14,NLIN,NLIN)
      CALL NATFRQ(EVR,EVI,WN,DAMPR,NLIN,NPOLES)
C =========  PRINT EIGENVALUES
      WRITE(6,2867) NLIN
2867  FORMAT(////45H   SYSTEM EIGENVALUES AT THIS OPERATING POINT
     1              ///28X,I3,2X,*EIGENVALUES*/13X,*REAL*,9X,*IMAGINARY*
     I, 6X,*NATURAL FREQ.*,5X,*DAMPING RATIO*)
      DO 2868 I=1,NPOLES
      J=IBLNK
      IF(EVI(I).GT.0.) J=IPOM
 2868 WRITE(6,2869)I,EVR(I),J,EVI(I),WN(I),DAMPR(I)
 2869 FORMAT(3X,I3,3X,G12.6,2X,A2,G12.6,4X,2G16.6)
      WRITE(6,2871)
2871  FORMAT(/////)
      GO TO 2783
C ========  ADVANCE PLOT INDEX AND SS PARAMETER
 2706 INDEX=INDEX+1
      PARAM=PARAM+DELTA
      GO TO 3000
 4100 WRITE(6,4101) INDEP
 4101 FORMAT(//10X,31H*** WARNING ***  CAN*T IDENTIFY,1X,A8,1X,
     1 33HAS A VALID STEADY STATE PARAMETER//)
      WRITE(6,2871)
      RETURN R1
      END
```

```
CSTABMX
      SUBROUTINE STABMX(NSIM,XDOTO,ICOUNT,RATIO,A,NLIN,ITEST)
C  VERSION 3.                          REVISED: APRIL 30 1976
C   PURPOSE  CALCULATE STABILITY MATRIX
C  CALL SEQUENCE
C   ITEST  = MODE OF STABILITY MATRIX CALCULATION
C            ITEST = 0  SKIP RATIO MATRIX CALCULATION
C            ITEST = 1  CALCULATE RATIO MATRIX + STABILITY MATRIX
C            ITEST = 2  CALCULATE STABILITY MATRIX + COMPARE WITH
C                       PREVIOUSLY CALCULATED A MATRIX
C            ITEST = 3  CALCULATE ONLY THOSE ELEMENTS OF A MATRIX
C                       AS INDICATED BY THE CODES STORED IN RATIO
C            ITEST = -1  SAME AS ITEST = 0.  (THIS CONDITION OCCURS
C                       WHEN MORE THAN 400 /A/ ELEMENTS CHANGE DURING
C                       ITEST = 2.)
      REAL XDOTO(1),RATIO(1),A(1),X(1),XDOT(1)
      INTEGER INT(1)
      COMMON /CX/X/CXDOT/XDOT/CINT/INT/CXIC/XIC(1)/ERMESS/IFATAL,IERR
      COMMON/CNTRLS/ANTYPE,IPRINT,MODE,ERROR(1)/CTIME/TIME
      EQUIVALENCE(RAT,IRAT)
      INDEXF(I1,I2,M1)=I1+(I2-1)*M1
C ========  SET X = XIC  (OPERATING POINT)
      DO 2810 I=1,NSIM
 2810 X(I)=XIC(I)
      IF(ITEST.GE.2) GO TO 2813
C =========  TURN ON ALL INTEGRATORS FOR INITIAL FUNCTION EVAL.
      DO 2812 I=1,NSIM
      IF(INT(I).EQ.0) INT(I)=2
 2812 CONTINUE
C =========  EVALUATE MODEL RATES
C ----------  TURN ON ERROR MESSAGES IN MODEL
2813  IERR=1
      CALL EQMO(TIME,TIME,1)
C ----------  TURN OFF ERROR MESSAGES IN MODEL
      IERR=0
C =========  SAVE NOMINAL MODEL RATES
      DO 2815 I=1,NSIM
 2815 XDOTO(I)=XDOT(I)
      IF(ITEST-2) 2816,2818,2900
 2816 NLIN=NSIM
C =========  RESTORE FROZEN STATES AFTER INITIAL FUNCTION EVAL. AND
C              DETERMINE THE ORDER OF THE MODEL
      DO 2814 I=1,NSIM
      IF(INT(I).NE.2) GO TO 2814
      INT(I)=0
      NLIN=NLIN-1
 2814 CONTINUE
 2818 K1=0
      ICOUNT=0
C ===============  SCAN COLUMNS OF STABILITY MATRIX  ==========
      DO 2845 J=1,NSIM
C =========  SKIP FROZEN STATES
      IF(INT(J).NE.0) GO TO 2820
      K1=K1+1
      GO TO 2845
C ========  PERTURB THE JTH STATE
 2820 X(J)=X(J)+ERROR(J)
C ========  EVALUATE MODEL RATES
```

```
      CALL EQMO(TIME,TIME,1)
      K2=0
C ================  SCAN ROWS OF STABILITY MATRIX  ============
      DO 2830 I=1,NSIM
C =========  SKIP FROZEN STATES
      IF(INT(I).NE.0) GO TO 2821
      K2=K2+1
      GO TO 2830
 2821 J2=INDEXF(I-K2,J-K1,NLIN)
      IF(ITEST.NE.2) GO TO 2825
C ==========  CALCULATE STABILITY MATRIX ELEMENT
      XDUM=(XDOT(I)-XDOTO(I))/ERROR(J)
C =========  COMPARE TO PREVIOUSLY CALCULATED VALUE
      IF(XDUM.EQ.A(J2)) GO TO 2830
      ICOUNT=ICOUNT+1
      IF(ICOUNT.GT.400) GO TO 2823
C ========  GENERATE CODE IDENTIFYING THOSE ELEMENTS THAT CHANGE
      IRAT=I+1000*J+1000000*J2
      RATIO(ICOUNT)=RAT
      A(J2)=XDUM
      GO TO 2830
 2823 ITEST=-1
C =========  CALCULATE STABILITY MATRIX ELEMENT
 2825 A(J2)=(XDOT(I)-XDOTO(I))/ERROR(J)
 2830 CONTINUE
      IF(ITEST.NE.1) GO TO 2846
C =========  LINEARITY EVALUATION USING 1/2 SIZE PERTURBATION
      X(J)=XIC(J)+.5*ERROR(J)
      CALL EQMO(TIME,TIME,1)
      K2=0
      DO 2644 I=1,NSIM
      IF(INT(I).NE.0) GO TO 2835
      K2=K2+1
      GO TO 2844
 2835 XDUM=(XDOT(I)-XDOTO(I))/ERROR(J)*2.
      J2=INDEXF(I-K2,J-K1,NLIN)
      IF(A(J2))2837,2840,2837
 2837 RATIO(J2)=XDUM/A(J2)
      IF(ABS(RATIO(J2)-1.).GT.    .1)  ICOUNT=ICOUNT+1
      GO TO 2844
 2840 RATIO(J2)=1.
 2844 CONTINUE
 2846 CONTINUE
      X(J)=XIC(J)
 2845 CONTINUE
      RETURN
C ========  CALCULATE ONLY THOSE ELEMENTS OF STABILITY MATRIX THAT
C              ARE CHANGING.
 2900 K1=ICOUNT
      K2=0
 2910 IF(K1.LE.0) RETURN
C ========  DETERMINE ELEMENTS OF STABILITY MATRIX TL BE EVALUATED
C             FROM CODES STORED IN RATIO.
      RAT=RATIO(K1)
      J2=IRAT/1000000
      J=IRAT/1000-J2*1000
      I=IRAT-J2*1000000-J*1000
      K1=K1-1
```

```
      IF(J.EQ.K2) GO TO 2920
      K2=J
      X(J)=X(J)+ERROR(J)
      CALL EQMO(TIME,TIME,1)
      X(J)=XIC(J)
C ========  CALCULATE STABILITY MATRIX ELEMENT
 2920 A(J2)=(XDOT(I)-XDOTO(I))/ERROR(J)
      GO TO 2910
      END
```

```
CSTEP1
      SUBROUTINE STEP1(TIME,TINC)
C  VERSION 4.                          REVISED: SEPT 17 1976
C       PURPOSE:  CALL INTEGRATION SCHEME SELECTED BY MODE VARIABLE
C  CALL SEQUENCE:  TIME  - CURRENT TIME
C                  TINC  - TIME STEP TO BE TAKEN TO NEXT REPORT INTERVAL
C  DESIGNED BY: J.D. BURROUGHS                        FEB 1974
      COMMON/CORDER/NSIM,MOV,MOP/CX/X(1)/CXDOT/XDOT(1)
      COMMON/CNTRLS/INSTR,IPRINT,MODE,ERROR(1)
     COMMON/CWORK/A(1)/CWORKN/NN,N1,N2,N3,N4,N5,N6,N7
      COMMON/CTIME/TIM/CSIMUL/DUM(7),TMAX/CNAMEX/NAMEX(1)
      COMMON/CDIFS/JSTART,KINIT,TP
C ============  SET NEXT PRINT TIME
      TP=TIME+TINC
  5   GO TO(500,100,600),MODE
C ======================  NRKVS INTEGRATOR  ====================
 100  CALL OVERLAY(5HNRKVS,4,1,6HRECALL)
      IF(TIME.GT.TMAX) WRITE(6,101) (I,NAMEX(I),A(N7+I-1),I=1,NSIM)
 101  FORMAT(//47X,*INTEGRATOR STEP SIZE LIMITING COUNTS*/
     1 5(I4,1X,A8,2H= ,G11.5))
      KINIT= 1
      IF(MODE.EQ.1.AND.TIME.LT.TP-.00001)GO TO 505
      RETURN
C -----          START GEAR INTEGRATION WITH INITIAL CALL TO NRKVS
 500  IF(KINIT.EQ.0) GO TO 100
C ====================  GEAR INTEGRATOR  =====================
505   CALL OVERLAY(6HNONSIM,4,2,6HRECALL)
      IF(KINIT.NE.0) RETURN
      GO TO 100
C ================  FIXED STEP INTEGRATOR  ==================
  600 DT2=TINC*.5
      CALL EQMO(TIME,TINC,0)
      DO 601 I=1,NSIM
      A(I)=X(I)+DT2*XDOT(I)
601   X(I)=X(I)+TINC*XDOT(I)
      TIME=TIME+TINC
      CALL EQMO(TIME,TINC,0)
      DO 602 I=1,NSIM
602   X(I)=A(I)+DT2*XDOT(I)
      RETURN
      END
```

```
CTABIN
      SUBROUTINE TABIN(TAB,TABNAM,MAXDIM,LOCTAB,NOTAB)
C  VERSION 2.1                          REVISED: JAN 7 1976
C  PURPOSE:  PROVIDE FREE FIELD READ OF TABULAR DATA FOR EITHER
C            SINGLE OR DOUBLE TABLE LOOKUPS
C  CALL SEQUENCE: TAB    - ARRAY INTO WHICH DATA WILL BE LOADED
C                 TABNAM - ARRAY OF ALLOWABLE TABLE NAMES
C                 MAXDIM - ARRAY OF MAX. DIMENSIONS FOR TABLES
C                 LOCTAB - ARRAY OF TABLE LOCATIONS IN ARRAY TAB
C                 NOTAB  - NO. OF TABLES IN MODEL
C  METHOD:  TABLE DESCRIPTION IS IN THE FOLLOWING FORMAT
C       CARD 1   TABLE   TABLE NAME    NX   NZ
C       CARD 2*  SECONDARY INDEPENDENT VARIABLE TABLE
C       CARD 3*  PRIMARY INDEPENDENT VARIABLE TABLE
C       CARD 4*  DEPENDENT VARIABLE TABLE
C          *USE AS MANY CARDS AS DESIRED. MUST START TABLE WITH
C           A NEW CARD. MUST GIVE NZ,NX, AND NX*NZ POINTS RESPECTIVELY
C           IN EACH TABLE.
C       NX - NO. OF POINTS IN PRIMARY IND. VAR. TABLE
C       NZ - NO. OF POINTS IN SECONDARY IND. VAR. TABLE
C       DATA ITEMS ARE FREE FIELD. ITEMS MUST BE SEPERATED BY EITHER
C       2 OR MORE BLANKS,COMMA,EQUALS, OR LEFT OR RIGHT PARENTHESIS
      COMMON/CIO/IREAD,IWRITE,IDIAG
      DIMENSION CARD(6),TAB(1),TABNAM(1),MAXDIM(1),LOCTAB(1)
10    NX=0
      NZ=0
      MODE=0
      WRITE(IWRITE,20)
20    FORMAT(////)
C --->      READ DATA CARD
100   READ(IREAD,101)CARD
101   FORMAT(8A10)
      IF(EOF(IREAD).NE.0)GO TO 6520
C --->      SET CHARACTER INDEX
      INDEX=1
C --->      LOCATE NEXT PHRASE
120   CALL NXTPH(CARD,INDEX,PHRS)
C --->      TEST FOR BLANK PHRASE
      IF(PHRS.EQ.10H           )GO TO 100
C --->      TEST OPERATING MODE
      IF(MODE.NE.0)GO TO 130
C ========================= MODE=0 == CHECK FOR  TABLE
      CALL NUMERC(PHRS),RETURNS(122)
      GO TO 100
122   IF(PHRS.NE.5HTABLE)GO TO 6500
      MODE=1
      GO TO 120
130   IF(MODE.GT.1)GO TO 140
C ========================= MODE=1 == STORE TABLE NAME
      CALL NUMERC(PHRS),RETURNS(160)
C --->      NUMERIC PHRS
      GO TO 6300
C --->      CONVERT BCD TO REAL
C ========================= MODE .GT. 1
140   CALL NUMERC(PHRS),RETURNS(6200)
      CALL BCDREL(PHRS,PHRS)
C --->      BRANCH TO TASK INDICATED BY MODE
160   GO TO(1000,2000,3000,4000,5000,6000),MODE
```

```
C ======================== MODE=1 == STORE TABLE NAME
1000  CALL LCMPH(PHRS,TABNAM,NOTAB,1,NTAB)
      IF(NTAB.LE.0)GO TO 1100
C --->      STARTING LOCATION FOR TABLE DATA
      LOC=LOCTAB(NTAB)
C --->      LAST WORD ADDRESS FOR TABLE DATA
      MAX=MAXDIM(NTAB)+LOC-1
      TAB(LOC)=PHRS
      MODE=2
      GO TO 120
1100  WRITE(IWRITE,1101)PHRS
1101  FORMAT(17H *** WARNING *** ,A10,
     1*IS NOT A VALID TABLE NAME FOR THIS MODEL.  DATA WILL BE IGNORED*)
      GO TO 10
C ======================== MODE=2 == STORE NO. POINTS IN PRI. IND. TABL
2000  TAB(LOC+1)=PHRS
      NXMAX=PHRS
      MODE=3
      CALL NXTPH(CARD,INDEX,PHRS)
      GO TO 140
C ======================== MODE=3 == STORE NO. POINTS IN SEC. IND. TABLE
3000  LOC=LOC+2
      TAB(LOC)=PHRS
      NZMAX=PHRS
C --->      TEST IF THERE IS A SECONDAY INDEPENDENT VAR. TABLE
      IF(NZMAX.LE.1) GO TO 3020
      MODE=4
      GO TO 3040
3020  MODE=5
      NZMAX=0
3040   ITAB=LOC
      IF(LOC+NXMAX+NZMAX+NXMAX*MAX0(1,NZMAX).LE.MAX)GO TO 100
      LIM=MAXDIM(NTAB)-3
      WRITE(IWRITE,3041)NXMAX,NZMAX,LIM
3041  FORMAT(17H *** WARNING *** ,I4,* PRIMARY AND *,I4,
     1* SECONDARY INDEPENDENT VARIABLE POINTS EXCEEDS THE *,
     2I4,* WORD STORAGE LIMIT FOR THE*/21X,
     3*FOLLOWING TABLE.    SOME DATA WILL BE LOST.*/)
      GO TO 100
C ==============  MODE=4 ==      STORE SECONDARY IND. VAR. TABLE
4000  NZ=NZ+1
      IF(NZ.GT.NZMAX)GO TO 4040
4020  ITAB=ITAB+1
C --->      LIMIT DATA TO TAB ARRAY MAX.
      IF(ITAB.LE.MAX)TAB(ITAB)=PHRS
      GO TO 120
4040  MODE=5
C ======================== MODE=5 == STORE PRI. IND. VAR. TABLE
5000  NX=NX+1
      IF(NX.LE.NXMAX)GO TO 4020
      MODE=6
      NX=0
      NZ=0
C ======================== MODE=6 == STORE DEPENDENT VAR. TABLE
6000  ITAB=ITAB+1
      IF(ITAB.LE.MAX)TAB(ITAB)=PHRS
      NX=NX+1
      IF(NX.LT.NXMAX)GO TO 120
```

```
      NX=0
      NZ=NZ+1
      IF(NZ.LT.NZMAX)GO TO 120
C --->      TABLE READ IN COMPLETE - PRINT
6020  WRITE(IWRITE,6021)TAB(LOC-2)
6021  FORMAT(20X,*TABLE *,A7/)
C --->      TEST IF THERE ARE 2 INDEPENDENT VAR.
      IF(NZMAX.LE.0)GO TO 6100
      WRITE(IWRITE,6031)
6031  FORMAT(10X,*SECONDARY INDEPENDENT VARIABLE TABLE*/)
      ITAB=LOC
      WRITE(IWRITE,6041)(TAB(ITAB+I),I=1,NZMAX)
6041  FORMAT(10(3X,G10.4))
6100  WRITE(IWRITE,6101)
6101  FORMAT(/10X,*PRIMARY INDEPENDENT VARIABLE TABLE*/)
      ITAB=LOC+NZMAX
      WRITE(IWRITE,6041)(TAB(ITAB+I),I=1,NXMAX)
      ITAB=LOC+NXMAX+NZMAX
      NZ=0
      WRITE(IWRITE,6121)
6121  FORMAT(/10X,*DEPENDENT VARIABLE TABLE*/)
6140  WRITE(IWRITE,6041)(TAB(ITAB+I),I=1,NXMAX)
      NZ=NZ+1
      IF(NZ.GE.NZMAX) GO TO 6400
      ITAB=ITAB+NXMAX
      GO TO 6140
6200  BACKSPACE IREAD
      WRITE(IWRITE,6201)CARD
6201  FORMAT(50H *** WARNING *** NON-NUMERIC DATA ON THIS CARD--->,8A10
     1/17X,*WILL READ NEXT TABLE*/)
      GO TO 6020
6300  WRITE(IWRITE,6301)CARD
6301  FORMAT(46H *** WARNING *** NON-ALPHA NAME ON THIS CARD--->,
     18A10/17X,*WILL IGNORE THIS CARD*/)
      GO TO 100
6400  WRITE(IWRITE,20)
      GO TO 10
6500  BACKSPACE IREAD
C --->      CHECK THAT ALL TABLES HAVE BEEN INPUT
6520  DO 6540 I=1,NOTAB
      LOC=LOCTAB(I)
      IF(KOMSTR(TABNAM(I),1,7,TAB(LOC),1).EQ.0)GO TO 6540
      WRITE(IWRITE,6531)TABNAM(I)
6531  FORMAT(//35H *** WARNING ***   DATA FOR TABLE  ,A7,
     1*  HAS NOT BEEN INPUT*/)
6540  CONTINUE
      RETURN
      END
```

```
CTFEVAL
      SUBROUTINE TFEVAL(OMEGA,POLES,POLE,N,R,LFLAG,IQUAD,PHASE)
C  VERSION 2.                          REVISED: DEC 23 1975
C  PURPOSE:  EVALUATE TRANSFER FUNCTION COMPLEX VALUE AT SPECIFIED
C            FREQUENCY.
C  CALL SEQUENCE:  OMEGA  - SPECIFIED FREQUENCY, R.M.S.  (COMPLEX)
C                  POLES  - SYSTEM EIGENVALUES WITH SM PARAMETER = 0
C                  POLE   - SYSTEM EIGENVALUES WITH NOMINAL SM PARAMETER
C                  N      - SYSTEM ORDER
C                  R      - OPEN LOOP TRANSFER FUNCTION  (COMPLEX)
C                  LFLAG  - ZERO PHASE FLAG.  LFLAG = 0 -- PHASE>TOL.
C                                             LFLAG = 1 -- PHASE<TOL.
C                  IQUAD  - TRANSFER FUNCTION QUADRANT
C                  PHASE  - APPROXIMATE PHASE ANGLE, RADIANS.
C  DESIGNED BY:  J.D. BURROUGHS                    FEB 1969
      COMPLEX POLES(N),POLE(N),OMEGA,R
      DOUBLE PRECISION RR,RI,NR,DR,DI,RR1,RR2,RR3,NI
      COMMON /CWORK/RR,RI,NR,DR,DI,RR1,RR2,RR3,NI
      RR=1.D0
      RI=0.D0
C ========  CALC. ONE MINUS THE COMPLEX PRODUCT OF (FREQ-CLOSED LOOP
C          POLES) / (FREQ-OPEN LOOP POLES)
      DO 100 I=1,N
      NR=DBLE(-REAL(POLE(I)))
      NI=DBLE(AIMAG(OMEGA)-AIMAG(POLE(I)))
      DR=DBLE(-REAL(POLES(I)))
      DI=DBLE(AIMAG(OMEGA)-AIMAG(POLES(I)))
      RR1=DABS(DR)
      RR2=DABS(DI)
      IF(RR1.EQ.0.D0.AND.RR2.EQ.0.D0)GO TO 100
      IF(RR1.LT.RR2)GO TO 50
   25 RR1=1.D0+(DI/DR)**2
      RR2=(NR/DR+NI*DI/DR**2)/RR1
      RR3=(-NR*DI/DR**2+NI/DR)/RR1
      RR1=RR*RR2-RI*RR3
      RI=RR*RR3+RI*RR2
      RR=RR1
      GO TO 100
   50 RR1=NR
      NR=NI
      NI=-RR1
      RR1=DR
      DR=DI
      DI=-RR1
      GO TO 25
  100 CONTINUE
      R=(1.,0.)-CMPLX(SNGL(RR),SNGL(RI))
      PHASE=1.570796
C ==========  CALCULATE APPROXIMATE PHASE ANGLE
      IF(REAL(R).NE.0.)PHASE=AIMAG(R)/REAL(R)
      LFLAG=0
C =========  TEST FOR ZERO PHASE.  TOLERANCE = .00001 RADIANS
      IF(ABS(PHASE).LT. .00001    )  LFLAG=1
C ========  DETERMINE QUADRANT OF TRANSFER FUNCTION
      IF(REAL(R).LT. 0.) GO TO 200
      IF(AIMAG(R).LT. 0.) GO TO 300
      IQUAD=1
      RETURN
```

```
200 IF(AIMAG(R).LT. 0.) GO TO 250
    IQUAD=2
    RETURN
250 IQUAD=3
    RETURN
300 IQUAD=4
    RETURN
    END
```

```
CTFBTCH
      OVERLAY(TFBTCH,7,0)
      PROGRAM TFBTCH
C  VERSION 3.1                        REVISED: OCT 7 1976
C  PURPOSE:  CONTROL THE CALCULATION OF TRANSFER FUNCTIONS.
C  DESIGNED BY: J.D. BURROUGHS                    FEB 1974
      COMMON/CORDER/NSIM,NOV,NOP
      COMMON/CPRON/DUM1(4),NINPUT,NOUT,DUM2(2)
      COMMON/CPROV/DUM3(11),FMAX,FMIN,DUM4(7)
      COMMON /CXIC/XIC(1 )/CINT/INT(1)
      COMMON /CNTRLS/ANTYPE,IPRINT,MODE,ERROR(1)
      COMMON /CWORK/GAIN(50),PHASE(50),FREQ(50),ITITLE(9),XMIN(2),
     1 XMAX(2),YMIN(2),YMAX(2)
      COMMON/CNAMEX/NAMEX(1)
      COMMON /CWORKN/NN,N1,N2,N3,N4,N5,N6,N7
      COMMON /CPLOTS/ INDPLT,INDWR,IOPT(30),PLOTID( 5),PTITLE( 8),
     +                IPOPT(10)
      REAL   XOPT(1)
      EQUIVALENCE (XOPT(1),IOPT(1))
      EQUIVALENCE(A,GAIN)
      REAL NAMEX,NINPUT,NOUT
      DIMENSION JTYPE(2),A(1)
      DATA JTYPE/1,3/
      DATA IBLNK /4H    /
       NPTS = 50
C ==========  GENERATE IDENTIFICATION CODES FOR  TF INPUT AND
C               TF OUTPUT  PARAMETERS
      CALL CODGEN(NINPUT,0,INPUT),RETURNS(4000)
      CALL CODGEN(NOUT,0,IOUT),RETURNS(4020)
      IOPT(3) = IBLNK
      IOPT(4) = IBLNK
      CALL DTTIM (IOPT(3))
      WRITE(6,3020)NINPUT,NOUT,PTITLE,IOPT(3),IOPT(4)
 3020 FORMAT(40X,48H/*/*/*/*/    TRANSFER FUNCTION ANALYSIS    /*/*/*/
     1 //51X,5HFROM ,A8,4H TO ,A8//26X,8A10//54X,2A12)
C ==========  PRINT OPERATING POINT AND PERTURBATION SIZE
      WRITE(6,2809)(I,NAMEX(I),XIC(I),ERROR(I),INT(I),I=1,NSIM)
 2809 FORMAT(//
     17X,5HSTATE,6X,*OPERATING   PERTURBATION   INTEGRATOR*/
     28X,*NAME*,8X,*POINT*,9X,*SIZE*,8X,*CONTROL*/
     2 (I5,1X,A10,G14.5,G12.3,I9))
C =========  CALCULATE TRANSFER FUNCTION GAIN AND PHASE
      CALL TRNFCN(INPUT,IOUT,NSIM,NLIN,GAIN,PHASE,FREQ,A,A(NN),A(N1),
     1 A(N2),A(N3),A(N4),A(N5),A(N6),A(N3),A(N4),A(N6),A(N1)),
     1 RETURNS(3390)
      JPLOT=1
C ==========  PRINT RESULTS IF MANUAL SCALES WERE NOT SPECIFIED.
      IF(FMIN.GE.FMAX)WRITE(6,3320)((FREQ(I+(K-1)*10),I=1,10),
     1 (GAIN(I+(K-1)*10),I=1,10),(PHASE(I+(K-1)*10),I=1,10),K=1,5)
 3320 FORMAT(/(*0 FREQ.,RPS:*,10G12.5/7X,*GAIN:*,10G12.5/6X,*PHASE:*,
     1 10G12.5))
      IF ( FMIN .GE. FMAX ) GO TO 400
      FREQ(1) = FMIN
      FREQ(NPTS) = FMAX
C ==========  CALCULATE RESPONSE OVER SPECIFIED RANGE
      CALL RESPON(INPUT,IOUT,NSIM,NLIN,GAIN,PHASE,FREQ,A,A(NN),A(N1),
     1 A(N2),A(N3),A(N4),A(N5),A(N6),A(N3),A(N4),A(N6),A(N1)),
     1 RETURNS(3390)
```

```
      WRITE(6,3320)((FREQ(I+(K-1)*10),I=1,10),
     1 (GAIN(I+(K-1)*10),I=1,10),(PHASE(I+(K-1)*10),I=1,10),K=1,5)
  400 CONTINUE
      WRITE(6,3321)
3321  FORMAT(/////)
C
C     SET PLOT PARAMETERS.
C
      IF ( INDPLT .EQ. 0 ) GO TO 6000
      IOPT(1) = 4
      IOPT(2) = IOPT(2) + 1
      IOPT(5) = IPOPT(5)
      IOPT( 6) = IPOPT(6)
      IOPT( 7) = IPOPT(7)
      IOPT( 8) = NPTS
      XOPT(9) = NINPUT
      XOPT(10) = NOUT
      DO 300 I=11,19
  300 IOPT(I) = 0
C ==========  WRITE PLOT DATA ONTO TAPE30
      WRITE (30) IOPT,PLOTID,PTITLE
      WRITE (30) (FREQ(I),I=1,NPTS),(GAIN(I),I=1,NPTS),
     + (PHASE(I),I=1,NPTS)
      INDWR = 1
      GO TO 6000
 3390 CONTINUE
      WRITE(6,3321)
      GO TO 6000
 4000 WRITE(6,4001) NINPUT
 4001 FORMAT(//10X,31H*** WARNING ***  CAN*T IDENTIFY,1X,A8,1X,
     1 44HAS A VALID TRANSFER FUNCTION INPUT PARAMETER//)
      CALL CODGEN(NOUT,0,IOUT),RETURNS(4020)
      WRITE(6,3321)
      GO TO 6000
 4020 WRITE(6,4021) NOUT
 4021 FORMAT(//10X,31H*** WARNING ***  CAN*T IDENTIFY,1X,A8,1X,
     1 45HAS A VALID TRANSFER FUNCTION OUTPUT PARAMETER//)
      WRITE(6,3321)
6000  CONTINUE
      END
```

```
CTHSB2
      SUBROUTINE THSB2(A,IC,IA,N,M)
C     SUBROUTINE TO TRANSFORM UPPER BLOCK TRIANGULAR MATRIX INTO
C     HESSENBURG FORM USING DIRECT REDUCTION WITH PIVOTING.  THE
C     MATRIX A IS REPLACED BY ITS UPPER HESSENBURG FORM AND THE
C     TRANSFORMATION ELEMENTS ARE STORED IN A BELOW THE FIRST
C     SUBDIAGONAL. PIVOT INFORMATION IS STORED IN IA.
C     IC IS THE VECTOR PASSING THE BLOCKING INFORMATION FROM THE
C     PRECONDITIONING ROUTINE PREC2.
C     METHOD USED IS ONE USING ELEMENTARY LINEAR TRANSFORMATIONS.
C
C     THIS PROGRAM WAS DESIGNED AND CODED BY A. FREDERICK FATH OF
C     BOEING COMPUTER SERVICES, SEATTLE, WASHINGTON.  THIS VERSION
C     WAS COMPLETED DURING APRIL 1975.
C
      DIMENSION A(N,1),IA(1),IC(1)
      DOUBLE PRECISION SUM
      DO 10 I=1,N
      IA(I)=I
 10   CONTINUE
      IF(N.LE.2) RETURN
      NA=IC(1)
      IB=1
      K=2
C     DETERMINE MAXIMUM SIGMA
 15   VM=0.
      IT=K
      IF(NA.LT.K+1) GO TO 32
      DO 17 I=K,NA
      IF(ABS(A(I,K-1)).LE.VM) GO TO 17
      VM=ABS(A(I,K-1))
      IT=I
 17   CONTINUE
C     INTERCHANGE ROWS AND COLUMNS
      J=IA(K)
      IA(K)=IA(IT)
      IA(IT)=J
      IF(K.EQ.IT) GO TO 22
      DO 20 J=1,N
      VM=A(K,J)
      A(K,J)=A(IT,J)
 20   A(IT,J)=VM
      DO 21 I=1,NA
      VM=A(I,K)
      A(I,K)=A(I,IT)
 21   A(I,IT)=VM
C     COMPUTE TRANSFORMATION VARIABLES
 22   IF(A(K,K-1)) 30,32,30
 30   KK=K+1
      VM=1./A(K,K-1)
      DO 31 J=KK,NA
 31   A(J,K-1)=A(J,K-1)*VM
C     COMPUTE NEW H AND SIGMAS
 32   K=K+1
      KK=K-1
      IK=1
      II=2
      IF(KK.LE.NA) GO TO 321
```

```
      IB=IB+1
      NA=IC(IB)
321   DO 36 I=1,NA
      IF(I.LE.IC(IK)) GO TO 322
      II=IC(IK)+2
      IK=IK+1
322   IF(K.GT.NA) GO TO 34
      SUM=A(I,KK)
      DO 33 J=K,NA
33    SUM=SUM+A(I,J)*A(J,KK-I)
      A(I,KK)=SUM
34    IF(I.LT.3) GO TO 36
      L=I-1
      IF(L.GT.KK) L=KK
      IF(II.GT.L) GO TO 36
      SUM=A(I,KK)
      DO 35 J=11,L
35    SUM=SUM-A(I,J-1)*A(J,KK)
      A(I,KK)=SUM
36    CONTINUE
      IF(K-N) 15,32,50
50    RETURN
      END
```

```
CTITLE
      SUBROUTINE TITLE (CARD,IN,TITL,NT)
C  VERSION 1.                              REVISED: MAY 15 1975
C
C     PURPOSE - TO LOCATE AND CENTER A TEXTUAL TITLE.
C
C        CARD - INPUT CARD IMAGE
C          IN - CHARACTER AT WHICH TO START SEARCH
C        TITL - RESULTING TITLE
C          NT - NUMBER OF CHARACTERS IN TITLE FIELD
C
      DIMENSION CARD(1),TITL(1)
      DATA BLNK /1OH           /
      DATA COMMA /1OH,          / , EQUAL /1OH=          /
C
C     FIND FIRST NON-BLANK CHARACTER.
C
      DO 10 I=IN,80
      I1 = I
      CALL GETT(CARD,I,CHAR)
      IF ( CHAR .EQ. COMMA ) GO TO 10
      IF ( CHAR .EQ. EQUAL ) GO TO 10
      IF ( CHAR .NE. BLNK ) GO TO 20
   10 CONTINUE
      RETURN
   20 CONTINUE
C
C     FIND LAST CHARCTER.
C
      I2 = 81
      DO 30 I=IN,80
      I2 = I2 - 1
      CALL GETT(CARD,I2,CHAR)
      IF ( CHAR .NE. BLNK ) GO TO 40
   30 CONTINUE
   40 CONTINUE
C
C     MOVE TITLE INTO TITL ARRAY.
C
      NW = (NT-1) / 10 + 1
      DO 50 I=1,NW
      TITL(I) = BLNK
   50 CONTINUE
      NC = I2 - I1 + 1
      J1 = (NT-NC) / 2 + 1
      J2 = J1 + NC - 1
      K = I1
      DO 60 I=J1,J2
      CALL GETT(CARD,K,CHAR)
      CALL PUTT(TITL,I,CHAR)
      K = K + 1
   60 CONTINUE
      RETURN
      END
```

```
CTRNFCN
      SUBROUTINE TRNFCN(INPUT,IOUT,NSIM,NLIN,GAIN,PHASE,FREQ,
     1 A,RATIO,XDOTO,DWORK,IA,IB,IC,ID,EVR,EVI,POLES,POLES1),RETURNS(R1)
C  VERSION 3.1                        REVISED: OCT 5 1976
C------>PURPOSE:
C         CALCULATE THE TRANSFER FUNCTION FROM SPECIFIED INPUT TO
C         OUTPUT POINTS IN MODEL.
C  CALL SEQUENCE:  INPUT  - TF INPUT IDENTIFICATION CODE
C                  IOUT   - TF OUTPUT IDENTIFICATION CODE
C                  NSIM   - NONLINEAR SYSTEM ORDER
C                  NLIN   - LINEAR MODEL ORDER  (WITHOUT FROZEN STATES)
C                  GAIN   - 50 X 1  GAIN ARRAY
C                  PHASE  - 50 X 1  PHASE ANGLE ARRAY
C                  FREQ   - 50 X 1  FREQUENCY ARRAY
C               NAME            DESCRIPTION              LOCATION
C                  A       - NLIN X NLIN  WORK SPACE     /CWORK/A(1)
C                  RATIO   - NLIN X NLIN  WORK SPACE     /CWORK/A(NN1
C                  XDOTO   - NLIN X 1   WORK SPACE       /CWORK/A(N1)
C                  DWORK   - NLIN X 1   WORK SPACE       /CWORK/A(N2)
C                  IA      - NLIN X 1   WORK SPACE       /CWORK/A(N3)
C                  IB      - NLIN X 1   WORK SPACE       /CWORK/A(N4)
C                  IC      - NLIN X 1   WORK SPACE       /CWORK/A(N5)
C                  ID      - NLIN X 1   WORK SPACE       /CWORK/A(N6)
C                  EVR     - NLIN X 1   WORK SPACE       /CWORK/A(N3)
C                  EVI     - NLIN X 1   WORK SPACE       /CWORK/A(N4)
C                  POLES   - NLIN X 1   WORK SPACE       /CWORK/A(N6)
C                  POLES1  - NLIN X 1   WORK SPACE       /CWORK/A(N1)
C            RETURN R1  --  RETURN TAKEN WHEN ALGEBRAIC LOOP EXISTS
C                           BETWEEN TRANSFER FUNCTION INPUT AND OUTPUT.
C  DESIGNED BY: J.D. BURROUGHS                          FEB 1974
      COMMON/CNTRLS/ANTYPE,IPRIN,IMODE,ERROR(1)/CTIME/TIME
      COMMON /CX/X(1)/CXDOT/XDOT(1)/CINT/INT(1)/CXIC/XIC(1)/CP/P(1)
      DIMENSION DWORK(1),IA(1),IB(1),IC(1),ID(1)
      DIMENSION POLES(1),POLES1(1),A(1),RATIO(1)
      REAL GAIN(1),PHASE(1),EVR(1),EVI(1),FREQ(1),XDOTO(1)
      COMPLEX POLES,POLES1,TRANS,OMEGA
      INDEX2(I,J,M)=I+(J-1)*M
      NPTS=50
C----->CALCULATE STABILITY MATRIX AT OP. POINT
      CALL STABMX(NSIM,XDOTO,ICOUNT,RATIO,A,NLIN,0)
C------>CALCULATE EIGENVALUES AND LOAD INTO COMPLEX ARRAY
      CALL EGVL3(A,RATIO,EVR,EVI,IA,IB,IC,ID,DWORK,1.E-14,NLIN,NLIN)
      DO 100 I=1,NLIN
  100 POLES(I)=CMPLX(EVR(I),EVI(I))
      IAL=0
C------>UPDATE XDOT
      CALL EQMO(TIME,TIME,1)
C------->DETERMINE I-O VALUES AT OP. POINT
      CALL VAROUT(IOUT,OUTO)
      CALL VAROUT(INPUT,AINPUT)
C------>PERTURB INPUT
      AIN=AINPUT-.1
C------>INPUT = PERTURBED VALUE (EVALUTE EFFECT ON MODEL)
      CALL SETIN(INPUT,AIN)
      CALL VAROUT(IOUT,OUT)
C------>RESTORE INPUT TO AINPUT
      CALL VARMOD(INPUT,AINPUT)
C------>PRINT WARNING
```

```
      IF(OUTO  .EQ.OUT) GO TO 350
      ALGAIN=(OUTO-OUT)*10.
      IAL=1
      WRITE(6,255) ALGAIN
  255 FORMAT(//10X,46H *** WARNING ***  ALGEBRAIC LOOP WITH GAIN OF  ,
     1 G12.6,* EXISTS BETWEEN INPUT AND OUTPUT*/
     2 *    THIS TRANSFER FUNCTION CAN NOT BE DETERMINED*)
      RETURN R1
C----->INACTIVE STATE COUNTER
  350 K1=0
C------>CALCULATE  A  WITH I/O LOOP OF -1 GAIN CLOSED
      DO 700 J=1,NSIM
      IF(INT(J).NE.0) GO TO 450
      K1=K1+1
C------>SCAN STATES
      GO TO 700
C------->PERTURB STATE
  450 X(J)=XIC(J)+ERROR(J)
C------>EVALUATE MODEL
      CALL EQMO(TIME,TIME,1)
C------>READOUT I/O
      CALL VAROUT(IOUT,OUTPUT)
      CALL VARCUT(INPUT,AINPUT)
      AIN=AINPUT-OUTPUT+OUTO
C------>MODIFY INPUT BY -1*  ANY CHANGE THAT HAS BEEN OCCURRED IN OUTPUT,
C------>AND EVALUATE MODEL
      CALL SETIN(INPUT,AIN)
C------>INACTIVE STATE COUNTER
      K2=0
C------>SCAN RATES
      DO 600 I=1,NSIM
      IF(INT(I).NE.0) GO TO 500
      K2=K2+1
      GO TO 600
C------>CALCULATE  A  ELEMENT
  500 A(INDEX2(I-K2,J-K1,NLIN))=(XDOT(I)-XDOTO(I))/ERROR(J)
  600 CONTINUE
C------>RESTORE INPUT TO AINPUT
      CALL VARMOD(INPUT,AINPUT)
C------>RESTORE STATE TO OP. POINT VALUE
      X(J)=XIC(J)
  700 CONTINUE
C------>CALCULATE EIGNEVALUES WITH -1 CLOSED AND LOAD INTO COMPLEX
C       ARRAY POLES1
      CALL EGVL3(A,RATIO,EVR,EVI,IA,IB,IC,XDOTO,DWORK,1.E-14,NLIN,NLIN)
      DO 750 I=1,NLIN
  750 POLES1(I)=CMPLX(EVR(I),EVI(I))
      FREQ(50)=-1.E36
      FREQ(1)=1.E36
C                *** DETERMINE FREQUENCY RANGE TO SCAN ***
      DO 800 I=1,NLIN
      IF(CABS(POLES(I)).LT.FREQ(1)) FREQ(1)=CABS(POLES(I))
      IF(CABS(POLES(I)).GT.FREQ(50)) FREQ(50)=CABS(POLES(I))
  800 CONTINUE
      FREQ(50)=10.*FREQ(50)
      FREQ(1)=.1*FREQ(1)
C              *** ENTRY POINT FOR MANUALLY SELECTED FREQ. RANGE ***
      ENTRY RESPON
```

```
      IF(FREQ(1).LE.0.) FREQ(1)=.01
      RAT=EXP(ALOG(ABS(FREQ(50)/FREQ(1)))/(NPTS-1))
      OMEGA=CMPLX(0.,FREQ(1))
C   CALC. FREQ. RESPONSE FROM NOMINAL AND MODIFIED SYSTEM POLES ***
      DO 1000 I=1,NPTS
      TRANS=(1.,0.)
      FREQ(I)=AIMAG(OMEGA)
      DO 900 J=1,NLIN
  900 TRANS=(OMEGA-POLES1(J))/(OMEGA-POLES(J))*TRANS
      TRANS=TRANS-(1.,0.)
      GAIN(I)=CABS(TRANS)
      IF(REAL(TRANS).EQ.0..AND.AIMAG(TRANS).EQ.0.) GO TO 950
      PHASE(I)=57.29578*ATAN2(AIMAG(TRANS),REAL(TRANS))
      GO TO 1000
  950 PHASE(I)=0.
 1000 OMEGA=RAT*OMEGA
      RETURN
      END
```

```
CVALUES
      SUBROUTINE VALUES(IPHRS,NAME,NO,VALUE,ITNO,MODE)
C  PURPOSE:  LOADS NUMERIC VALUES OF QUANTITIES IDENTIFIED BY DEFINE
C            STATEMENTS.
C  CALL SEQUENCE:  IPHRS = ARRAY CONTAINING NEXT PHRASE TO BE EXAMINED
C                  NAME  = ARRAY CONTAINING NAMES OF DEFINED QUANTITIES.
C                  NO     = NUMBER OF DEFINED QUANTITIES.
C                  VALUE = ARRAY INTO WHICH NUMERIC VALUES ARE TO BE LOA
C                  ITNO  = POSITION OF GIVEN QUANTITY IN NAME ARRAY.
C                  MODE  = MODE OF OPERATION.
C                  MODE  = 0  A NAME CAN*T BE IDENTIFIED.
C                  MODE  = 2  NAME HAS BEEN IDENTIFIED.
      DIMENSION NAME(NO),VALUE(NO)
      REAL IPHRS,NAME
C  TEST FOR NUMERIC FIRST CHARACTER.
      CALL NUMERC(IPHRS),RETURNS(50)
      GO TO 200
C  SEARCH NAMELIST FOR NAME CONTAINED IN IPHRS.
50    CALL LCMPH(IPHRS,NAME,NO,1,ITNO)
      IF(ITNO.LE.0) GO TO 100
C  NAME FOUND AT LOCATION ITNO.
      MODE=2
      RETURN
C  NAME NOT FOUND.
100   WRITE(6,101) IPHRS
101   FORMAT(15X,33H*** WARNING ***  CAN*T IDENTIFY  ,A10,
     1 23H  VALUE WILL BE IGNORED)
      MODE=-1
      RETURN
C  TEST MODE TO ASSURE THAT NAME HAS BEEN IDENTIFIED.
200   IF(MODE.NE.2) GO TO 300
C  CONVERT NUMERIC VALUE CONTAINED IN IPHRS FROM A TO G FORMAT.
      CALL BCDREL(VALUE(ITNO),IPHRS)
      MODE=0
      RETURN
300   WRITE(6,301)IPHRS
301   FORMAT(15X,71H*** WARNING ***  A VALID PARAMETER NAME MUST PRECEDE
     1 THE NUMERIC VALUE:  ,A10)
      RETURN
      END
```

```
CVARMOD
      SUBROUTINE VARMOD(I,VAR)
C  PURPOSE:  TO MODIFY THE CURRENT VALUE OF A STATE,VARIABLE,
C            PARAMETER, ETC. GIVEN THE INTEGER IDENTIFICATION CODE
C            FOR THE QUANTITY.
C  CALL SEQUENCE:  I    = IDENTIFICATION CODE.
C                  VAR = NEW NUMERIC VALUE BEING INPUT.
      COMMON/CX/X(1)/CXDOT/XDOT(1)/CV/V(1)/CP/P(1)/CXIC/XIC(1)
      COMMON/CTIME/TIME
C  TEST FOR PARAMETER CODE
      IF(I.LE.4000000.OR.I.GT.5000000) GO TO 10
      P(I-4000000)=VAR
      RETURN
C  TEST FOR IC CODE
10    IF(I.LE.2000000.OR.I.GT.3000000) GO TO 20
      XIC(I-2000000)=VAR
      RETURN
C  TEST FOR VARIABLE CODE
20    IF(I.LE.3000000.OR.I.GT.4000000) GO TO 30
      V(I-3000000)=VAR
      RETURN
C  TEST FOR STATE CODE
30    IF(I.LT.1.OR.I.GT.1000000) GO TO 40
      X(I)=VAR
      RETURN
C  TEST FOR RATE CODE
40    IF(I.LE.1000000.OR.I.GT.2000000) GO TO 50
      XDOT(I-1000000)=VAR
      RETURN
C  TEST FOR TIME CODE
50    IF(I.EQ.0) TIME=VAR
      RETURN
      END
```

```
CVAROUT
      SUBROUTINE VAROUT(I,VAR)
C  PURPOSE:  TO RETRIEVE THE NUMERIC VALUES OF STATES,VARIABLES,
C            PARAMETERS,ETC. GIVEN THE INTEGER IDENTIFICATION CODE
C            FOR THE QUANTITY DESIRED.
C  CALL SEQUENCE:  I   = IDENTIFICATION CODE.
C                  VAR = NUMERIC VALUE RETURNED.
      COMMON/CX/X(1)/CXDOT/XDOT(1)/CV/V(1)/CP/P(1)/CXIC/XIC(1)
      COMMON/CTIME/TIME
C  TEST FOR TIME CODE
      IF(I.NE.0) GO TO 10
      VAR=TIME
      RETURN
C  TEST FOR STATE CODE
10    IF(I.LT.1.OR.I.GT.1000000) GO TO 20
      VAR=X(I)
      RETURN
C  TEST FOR VARIABLE CODE
20    IF(I.LE.3000000.OR.I.GT.4000000) GO TO 30
      VAR=V(I-3000000)
      RETURN
C   TEST FOR RATE CODE
30    IF(I.LE.1000000.OR.I.GT.2000000) GO TO 40
      VAR=XDOT(I-1000000)
      RETURN
C   TEST FOR PARAMETER CODE
40    IF(I.LE.4000000.OR.I.GT.5000000) GO TO 50
      VAR=P(I-4000000)
      RETURN
C  TEST FOR IC CODE
50    IF(I.LE.2000000.OR.I.GT.3000000) GO TO 60
      VAR=XIC(I-2000000)
      RETURN
C   CODE NOT IDENTIFIED.  SET VAR TO LARGE NUMBER.
60    VAR=1.E36
      RETURN
      END
```

```
CXFR
      SUBROUTINE XFR(X,Y,N)
      DIMENSION X(N),Y(N)
      DO 100 I=1,N
  100 Y(I)=X(I)
      RETURN
      END
```

4.0 PERMANENT FILE MAINTENANCE PROGRAM DESCRIPTION

4.1 INTRODUCTION

The Permanent File Maintenance program (FILOAD) is used to load and modify standard component input-output descriptions which are kept on the permanent file, WMPF. This program is used only when it is necessary to modify the input, output, or table list of an existing standard component or when a new standard component is to be added to the system.

4.2 PROGRAM STRUCTURE

Figure 4.2-1 contains a macro flow diagram of the Permanent File Maintenance program. Statement numbers in the main (FILOAD) program are given for each of the program's five principle tasks. The sequence of performing these tasks depends on the program commands. As each command is read it is printed on the lineprinter to provide a record of progress through the set of commands.

4.2.1 Command Interpretation

The command interpretation process for the FILOAD program is shown on Figure 4.2-2. Each phrase is tested against the five possible command phrases: LIST STANDARD COMPONENTS, PURGE, NEW FILE, DUMP FILE, and SYMBOL. If one of these phases is identified, branching occurs from statement 300 to a location that performs these tasks.

The LIST STANDARD COMPONENTS command sets a flag, (LIST=1), which causes the input, output, and table lists of any new or modified components to be printed upon the completion of processing all input commands. The PURGE command causes the name of the purged component to be removed from the list of standard component names, CMPNTS. This results in the removal of all name lists associated with that component from the WMPF file, when the degas process is performed at the end of the run. The SYMBOL command causes the symbol number

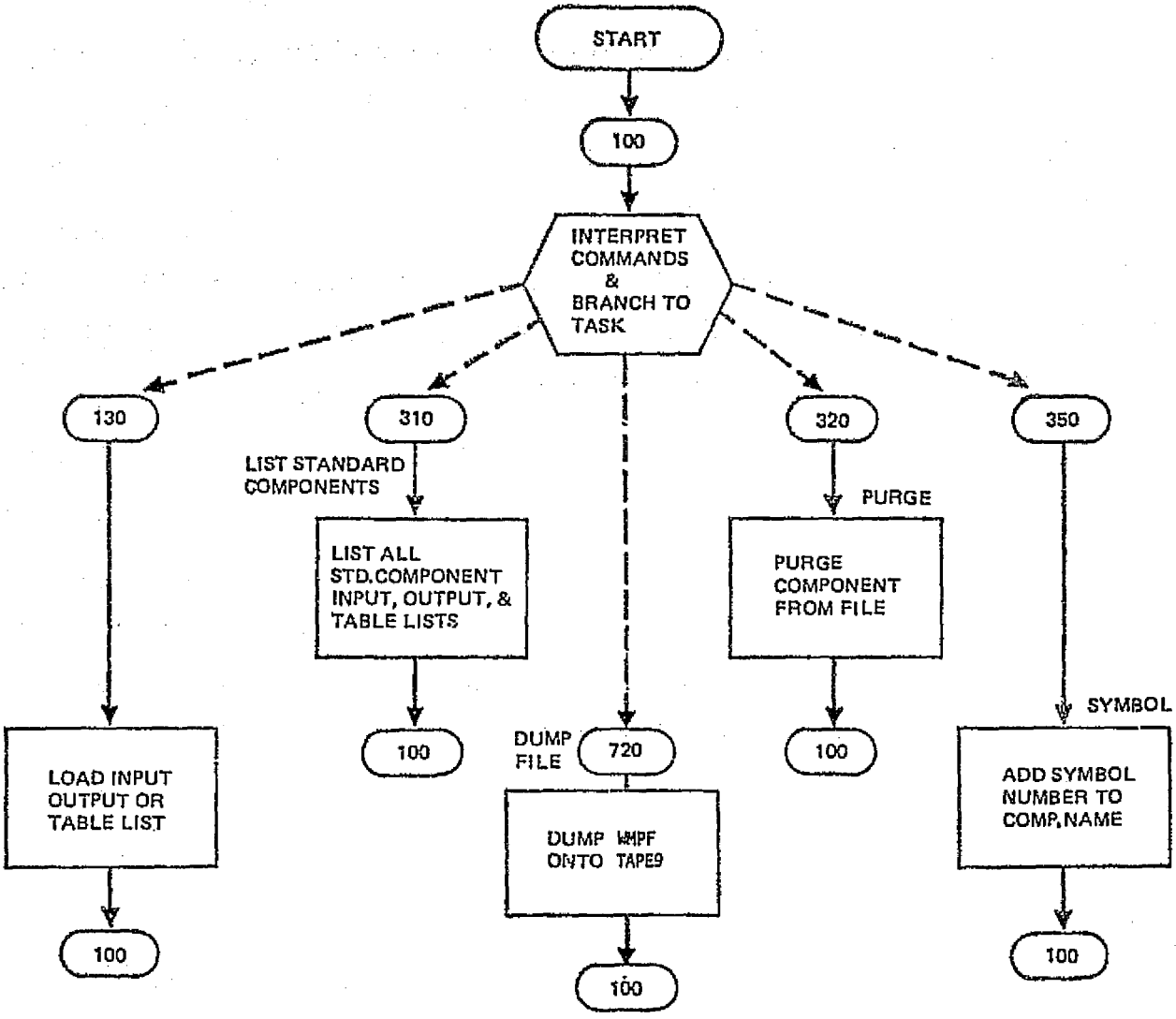

FIGURE 4.2-1 PERMANENT FILE MAINTENANCE PROGRAM - MACRO FLOW DIAGRAM

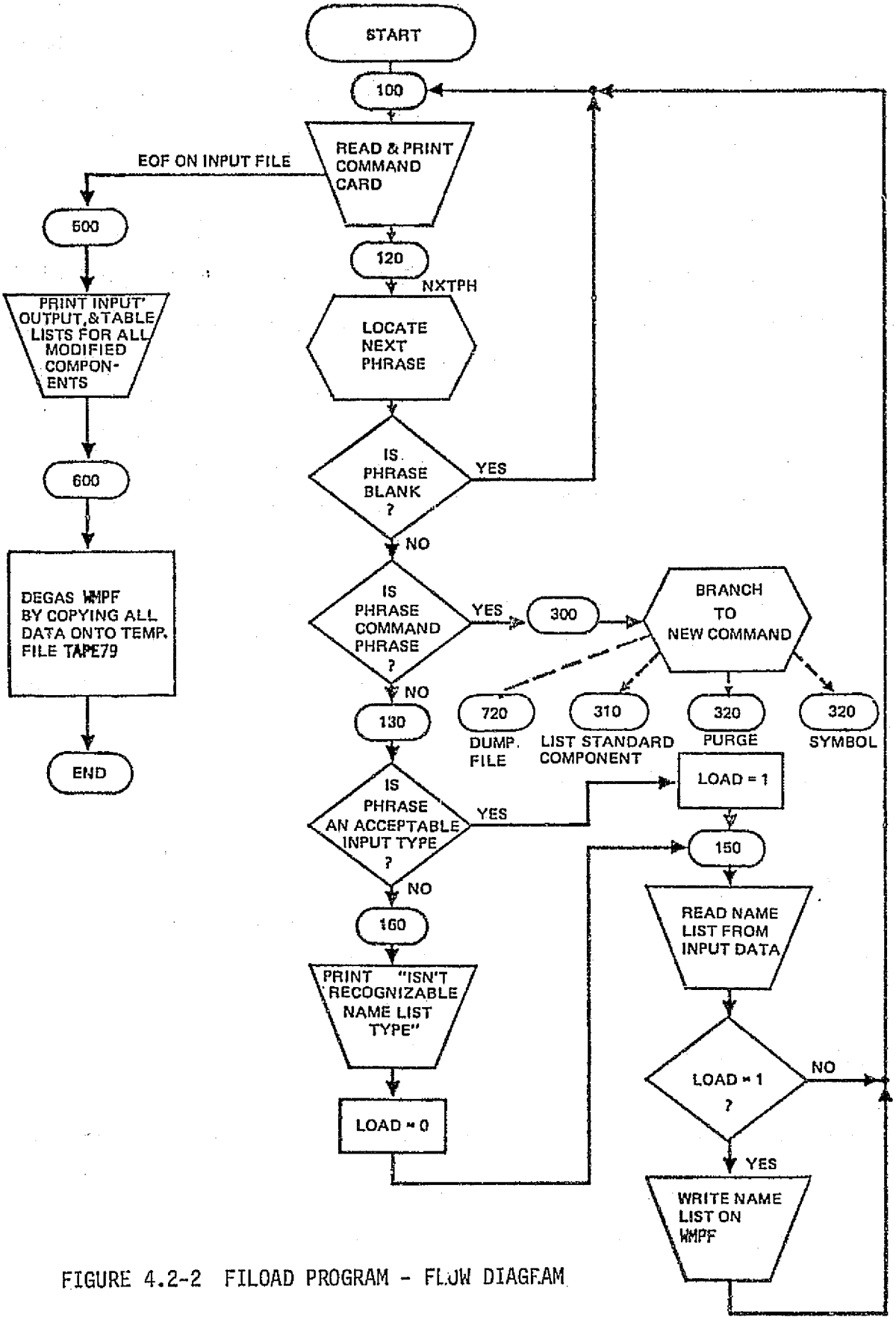

FIGURE 4.2-2 FILOAD PROGRAM - FLOW DIAGRAM

following a standard component name to be added to characters 9 and 10 of that name via the PUTCOD routine.

4.2.2 Name List Loading

If a phrase is not a command phrase, characters 3 through 6 are compared to the three acceptable input name list types: INPT, OUTP, and TABS. If one of these three types is not recognized, a warning message is printed and a flag (LOAD=0) is set to prevent data from being loaded onto the WMPF file. If a recognizable name list type occurs, the component name is obtained from characters 1 and 2 of the phrase. This component name is compared to existing component names. If it is an existing component name, the specified name list for that component is modified. If the component name does not match an existing component name, the new component name is added to the list of library components and a notice is printed that a new component has been added. Default input, output, and table name lists of zero length are then added to the WMPF file to assure that all three lists exist for all components. This is necessary to prevent READMS errors in the Model Generation program for components that might otherwise not have table name lists. The name list contained in the input data is then read and loaded onto the WMPF file.

The name list data is not in a free field format. The number of names must match that given in the phrase following the input list name, and the format of the name data must match that given in Section 7.0 of Volume I. Errors in formatting name list data can cause erroneous lists to be loaded. These will lead to errors in connections to the affected component.

4.2.3 File Degas Procedure

The WRITMS routine leaves previous versions of stored items on the permanent file as "dead space" whenever the new version is of a different length than the original. In order to remove this dead space, the FILOAD program creates a new copy of the WMPF file on local file TAPE79 upon the completion of each run.

TAPE79 is loaded by copying the input, output, and table name list for each component listed in the list CMPNTS, from WMPF. During this copy, the name lists for any purged components are deleted. Upon successful completion of the run, TAPE79 is copied onto WMPF.

4.2.4 Permanent Files

The random access permanent file WMPF is referred to in the FILOAD program as TAPE78. This file contains an input, output, and table name list for each standard component and a list of all standard component names.

4.2.5 Warning Messages

Table 4.2-1 lists the three warning messages that can be generated by the FILOAD program. These messages are preceded by: ***WARNING***. If either messages 1 or 2 are printed, the name list associated with these warnings will not be loaded. Other correct name lists for that or other components will be loaded.

4.3 FILOAD PROGRAM SOURCE LISTINGS

Compilation listings of the source code for the FILOAD program follows. Some of the subroutines are also used in the other programs. The names of the FILOAD routines, listed in alphabetical order, are:

BCDREL	KOMPAR
COMDAT	KOMSTR
CSORT	LCMPH
DUMPPF	NUMERC
FILOAD	NXTPH
GETCOD	PUTCOD
GETT	PUTT
ISCAN	STRMOV

TABLE 4.2-1 PERMANENT FILE MAINTENANCE PROGRAM
WARNING MESSAGES

1. CAN'T IDENTIFY xx AS A STANDARD COMPONENT

   The phase xx following the command PURGE or SYMBOL is not an existing standard component name. Check spelling of xx.

2. IN xxxxxxxxxx zzzz ISN'T A RECOGNIZED NAME LIST TYPE.
   NAME LIST WILL NOT BE LOADED.

   Characters 3 through 6, zzzz, in the phrase xxxxxxxxxx should be one of the name list types: INPT, OUTP, or TABS. Check spelling of xxxxxxxxxx.

3. xxxxxxxxxx ISN'T A VALID NUMBER OF NAMES FOR NAME LIST.
   NAME LIST WILL NOT BE LOADED.

   A numeric phrase giving the number of names in the following name list must follow the component name list type phrase.

```
CBCDREL
      SUBROUTINE BCDREL(VALUE,PHRS)
C  PURPOSE: CONVERT BCD NUMERIC INFORMATION INTO REAL FORMAT
C  CALL SEQUENCE:  VALUE - REAL NUMERIC VALUE ON RETURN
C                  PHRS - LEFT ADJUSTED BCD CHARACTERS ON INPUT.
C --->       SCAN CHARACTERS RIGHT TO LEFT TO LOCATE FIRST NON-BLANK
      DO 100 I=1,10
      J=11-I
      CALL GETT(PHRS,J,CHAR)
      IF(CHAR.NE.10H           ) GO TO 120
100   CONTINUE
C --->       CALC. NO. OF BITS OF LEFT CIRCULAR SHIFT REQ D
120   J=6*J
      VALUE=SHIFT(PHRS,J)
C --->       CONVERT BCD -> REAL
      DECODE(10,101,VALUE)VALUE
101   FORMAT(G10.0)
      RETURN
      END
```

```
CCOMDAT
      SUBROUTINE COMDAT(COMNAM,TYPE,N,NAMES)
C  PURPOSE:  OBTAIN LISTS OF INPUTS, OUTPUTS, OR TABLES REQUIRED
C            FOR A SPECIFIED STANDARD COMPONENT
C  CALL SEQUENCE:  COMNAM - STANDARD COMPONENT NAME
C                  TYPE   - TYPE OF LIST REQUESTED E.G. INPT,OUTP,TABS
C                  N      - NUMBER OF NAMES IN LIST
C                  NAMES  - NAMES OF QUANTITIES
C  METHOD:  LISTS ARE STORED ON A RANDOM ACCESS PERMANENT FILE AND
C           ACCESSED VIA THE MASS STORAGE I/O FEATURES OF FTN.
C         FOR EACH STANDARD COMPONENT, 3 LISTS WILL BE CREATED
C           WITH THE INDEX NAMES: XXINPT, XXOUTP, XXTABS  WHERE XX
C           REPRESENTS THE STANDARD COMPONENT NAME.  THE FIRST WORD
C           IN EACH LIST WILL CONTAIN THE NUMBER OF WORDS IN THE LIST
C           PLUS 1.
      COMMON/CIO/IREAD,IWRITE,IDIAG
      DIMENSION NAMES(1)
      COMMON/CMSI/MSI(1)
C --->      FORM INDEX
      AINDEX=10H
      CALL STRMOV(COMNAM,1,2,AINDEX,1)
      CALL STRMOV(TYPE,1,4,AINDEX,3)
C --->      READ FIRST WORD IN RECORD
      CALL READMS(78,N,1,AINDEX)
C --->      READ N WORDS
      IF(N.LT.1)N=1
      CALL READMS(78,NAMES,N,AINDEX)
      IF(N.LE.1) GO TO 200
C --->      SHIFT WORDS OVER ONE TO ELLIMINATE NO. OF WORDS STORED IN 1S
      DO 100 I=2,N
      NAMES(I-1)=NAMES(I)
100   CONTINUE
      N=N-1
      IF(IDIAG.EQ.80)WRITE(IWRITE,101)(NAMES(I),I=1,N)
101   FORMAT(* COMDAT-NAMES*/(6A10))
      RETURN
200   N=0
      IF(IDIAG.EQ.80)WRITE(IWRITE,201)
201   FORMAT(* COMDAT-N=0*)
      RETURN
      END
```

```
CCSORT
      SUBROUTINE CSORT (IA,NN)
C******
C     PURPOSE
C        CSORT SORTS THE ELEMENTS OF A SINGLE-DIMENSION SINGLE-
C        PRECISION ARRAY IN ASCENDING-CHARACTER (DISPLAY CODE) ORDER,
C        WITH A SORT OPTION THAT PLACES BLANK CHARACTERS FIRST IN THE
C        ALPHAMERIC SEQUENCE.
C        THE SHELL ALGORITH IS USED.
C     USAGE
C        DIMENSION IA(J)      WHERE J=IABS(N)
C        CALL CSORT(IA,N)
C     INPUT PARAMETERS
C           IA - INPUT ARRAY TO BE SORTED IN PLACE
C            N - IABS(N) IS NUMBER OF ELEMENTS IN ARRAY IA
C                N.LT.0  PERFORM NORMAL SORT, SEE ABSTRACT
C                N.GT.0  PERFORM MODIFIED SORT, SEE ABSTRACT
C     OUTPUT PARAMETERS
C           IA - THE INPUT ARRAY IS SORTED IN PLACE
C     USER ERROR
C        WHEN N.EQ.0, CONTROL IS RETURNED TO THE CALLING PROGRAM
C        WITHOUT SORTING.
C******
C
      INTEGER IA(1)
      EQUIVALENCE (ITEMP,KHR1)
      DATA IBLANK/55000000000000000008/
C * * *  VALIDITY CHECKS
      IF (NN.EQ.0) GO TO 990
      N = IABS(NN)
      IF (NN.LT.0) GO TO 300
C * * *  SWITCH CHARACTERS
      KHR1 = IBLANK
      KHR2 = 0
      LIM = 10*N
  200 K2 = 1
      DO 210 J=1,LIM
      N2 = LIM-K2+1
      I = ISCAN (KHR1,1,1,IA,K2,N2,M1)
      IF (I.EQ.0) GO TO 220
      CALL PUTT(IA,I,KHR2)
      IF (I.GE.LIM) GO TO 220
  210 K2 = I+1
  220 IF (KHR1.EQ.0) GO TO 990
C * * *  SORT THE ARRAY
  300 M = N
  320 M = M/2
      IF (M.LE.0) GO TO 400
      K = N-M
      DO 340 J=1,K
      I = J
  330 II = I+M
      CALL KOMPAR (IA(I),IA(II),ITEMP)
      IF (ITEMP.LE.0) GO TO 340
      ITEMP = IA(I)
      IA(I) = IA(II)
      IA(II) = ITEMP
```

```
      I = I-M
      IF (I.GT.0) GO TO 330
  340 CONTINUE
      GO TO 320
C * * *  SWITCH CHARACTERS BACK
  400 IF (NN.LT.0) GO TO 990
      KHR1 = 0
      KHR2 = IBLANK
      GO TO 200
  990 RETURN
      END
```

```
CDUMPPF
      SUBROUTINE DUMPPF(CMPNTS,ICPMAX,MSI,TYPES,AINPUT)
C  VERSION 1.                          REVISED: MAY 21 1976
C  PURPOSE:  DUMP PERMANENT FILE ONTO TAPE 9 IN INPUT FORMAT
C  CALL SEQUENCE:  CMPNTS - COMPONENT NAME LIST
C                  ICPMAX - NUMBER OF COMPONENTS
C                  MSI    - MASS STORAGE INDEX ARRAY
C                  TYPES  - DATA TYPE NAMES
C                  AINPUT - NAME ARRAY WORK STORAGE ARRAY
C  DESIGNED BY: J.D. BURROUGHS                      DEC 1975
      DIMENSION CMPNTS(1),MSI(1),TYPES(3),AINPUT(1)
      WRITE(9,11)
11    FORMAT(*NEW FILE*)
C ---       LOAD FILE NAME
      CALL READMS(78,PFNAME,1,6HPFNAME)
      WRITE(9,65)PFNAME
65    FORMAT(*FILE NAME=*,A10)
C --->      SCAN ALL COMPONENTS
      DO 640 I=1,ICPMAX
C --->      LOAD COMPONENT NAME
      PINDEX=10H
      CALL STRMOV(CMPNTS(I),1,2,PINDEX,1)
C --->      SCAN THREE TYPES OF LISTS REQ D FOR EACH COMPONENT
      DO 620 J=1,3
      CALL STRMOV(TYPES(J),1,4,PINDEX,3)
C --->      READ LISTS FROM FILE 78
      CALL READMS(78,MAX,1,PINDEX)
      CALL READMS(78,AINPUT,MAX,PINDEX)
      MAXM1=MAX-1
C --->      WRITE INPUT LIST NAME AND NUMBER OF INPUTS (OUTPUTS)
      WRITE(9,101)PINDEX,MAXM1
101   FORMAT(A7,* = *,I4)
C --->      TEST FOR TABLE INPUTS
      IF(J.EQ.3)GO TO 200
C --->      INPUT AND OUTPUT LIST TYPES
      IF(MAX.GT.1)WRITE(9,111)(AINPUT(K),K=2,MAX)
111   FORMAT(8A10)
      GO TO 620
C --->      TABLE INPUT FORMAT
200   IF(MAX.LE.1)GO TO 620
      DO 240 K=2,MAX
      CALL GETCOD(5,AINPUT(K),IDIM)
      DIM=IDIM
C --->      WRITE TABLE NAME AND MAX. DIMENSION
      WRITE(9,201)AINPUT(K),DIM
201   FORMAT(A3,F7.0)
240   CONTINUE
620   CONTINUE
C --->      TEST FOR SYMBOL NUMBER
      IF(KOMSTR(CMPNTS(I),9,2,2H  ,1).EQ.0)GO TO 640
C --->      GET SYMBOL NUMBER FROM COMPONENT NAME
      CALL GETCOD(5,CMPNTS(I),ISYMB)
      WRITE(9,631)CMPNTS(I),ISYMB
631   FORMAT(*SYMBOL, *,A2,* = *,I5)
640   CONTINUE
      RETURN
      END
```

```
CFILOAD
      PROGRAM FILOAD(INPUT=100,OUTPUT=200,TAPE5=INPUT,TAPE6=OUTPUT,
     1 TAPE3,TAPE78,TAPE79,TAPE9)
C   VERSION 4.                              REVISED  JUNE 24 1977
C   PURPOSE:  THIS PROGRAM ADDS INPUT,OUTPUT,AND TABLE NAME LISTS
C             TO THE EASY PROGRAM PERMANENT FILE.
C   METHOD:  DATA IS READ FROM TAPE3 AND LOADED INTO THE PERMANENT FILE.
C             THE DATA FORMAT IS: FIRST PHRASE =RECORD NAME.
C                                 SECOND PHRASE = NO. WORDS IN RECORD
C                THE INPUT AND OUTPUT NAME LISTS INPUT
C                DATA IS FIXED FIELD WITH A 8A10 FORMAT.
C                THE TABLE LIST INPUT DATA IS A10,G5.0
C                FORMAT.
C                THE NUMERIC INPUT SPECIFIES THE MAXIMUM
C                TABLE DIMENSION.  NEGATIVE VALUES
C                INDICATE SINGLE INDEPENDENT VARIABLE TABLES.
C   DESIGNED BY: J.D.BURROUGHS                          MAY 1974
      DIMENSION NAMES(100),MSI(897),CMPNTS(151),AINPUT(63),OUTPUT(63),
     1 TABLE(16),MSI2(897),ICOM(8),TYPES(3),CMMNDS(6),ICMMOD(151)
      COMMON/CIO/IREAD,IWRITE,IDIAG
      EQUIVALENCE (ICMP1,CMPNTS)
      DATA COMNAM/10H          /,TYPES/30HINPT      OUTP      TABS
     1/
      DATA CMMNDS/60HLIST STANDPURGE     DUMP FILE SYMBOL    NEW FILE  F
     1ILE NAME /
      DATA TYPE/10H          /,LIST/0/,ICPMOD/0/,ICMMAX/6/,ICPMAX/-1/
      IREAD=5
      IWRITE=6
      IDIAG=0
C --->      OPEN MASS STORAGE FILE
      CALL OPENMS(78,MSI,897,1)
      CALL OPENMS(79,MSI2,897,1)
C --->      READ COMMAND CARD
100   READ(3,101)ICOM
101   FORMAT(8A10)
      IF(EOF(3))500,120
120   INDEX=1
C --->      LOCATE NEXT PHRASE
      CALL NXTPH(ICOM,INDEX,PINDEX)
      IF(PINDEX.EQ.10H          )GO TO 100
C --->      SEARCH COMMAND LIST
      CALL LCMPH(PINDEX,CMMNDS,ICMMAX,1,NTASK)
C --->      BRANCH TO 300 IF COMMAND IS IDENTIFIED
      IF(NTASK.NE.0)GO TO 300
C --->      TEST IF COMPONENT NAME LIST HAS BEEN READ
      IF(ICPMAX.LT.0)GO TO 400
C --->      GET LIST TYPE
130   CALL STRMOV(PINDEX,3,4,TYPE,1)
C --->      COMPARE TYPE TO 3 ACCEPTABLE TYPES
      CALL LCMPH(TYPE,TYPES,3,1,ITYPE)
C --->      TEST IF TYPE WAS IDENTIFIED
      IF(ITYPE.EQ.0)GO TO 160
      LOAD=1
C --->      GET COMPONENT NAME
      CALL STRMOV(PINDEX,1,2,COMNAM,1)
```

```
C ---        BYPASS SEARCH IF COMPONENT COUNT < 1
      IF(ICPMAX.LT.1)GO TO 136
C --->       SEARCH COMPONENT NAME LIST
      DO 132 NCOMP=1,ICPMAX
      IF(KOMSTR(CMPNTS(NCOMP),1,2,COMNAM,1).EQ.0)GO TO 140
132   CONTINUE
C --->       NEW COMPONENT
136   ICPMAX=ICPMAX+1
      NCOMP=ICPMAX
C --->       ADD COMPONENT NAME TO LIST
      CMPNTS(ICPMAX)=COMNAM
      WRITE(6,137)COMNAM
137   FORMAT(3X,A4,*WILL BE ADDED AS A NEW COMPONENT*)
C ---        LOAD NAME ARRAYS WITH DEFAULT VALUES OF 0 NAMES
      VALUE=COMNAM
      DO 138 I=1,3
C ---        ADD TYPE NAME TO COMPONENT NAME
      CALL STRMOV(TYPES(I),1,4,VALUE,3)
      NAMES(1)=1
      CALL WRITMS(78,NAMES,1,VALUE)
138   CONTINUE
C ---        BYPASS SEARCH IF MODIFIED COMPONENT COUNTER = 0
140   IF(ICPMOD.EQ.0)GO TO 146
C --->       TEST IF COMPONENT HAS BEEN MODIFIED BEFORE
      DO 144 I=1,ICPMOD
      J=ICMMOD(I)
      IF(KOMSTR(COMNAM,1,2,CMPNTS(J),1).EQ.0)GO TO 150
144   CONTINUE
146   ICPMOD=ICPMOD+1
C --->       ACCUMULATE COMP. NOS. OF COMPONENTS MODIFIED
      ICMMOD(ICPMOD)=NCOMP
C --->       GET NEXT PHRASE WHICH CONTAINS NO. OF ITEMS IN LIST
150   CALL NXTPH(ICOM,INDEX,PHRS)
C --->       TEST FOR NUMERIC FIRST CHARACTER
      CALL NUMERC(PHRS),RETURNS(180)
C --->       CONVERT HOLLORITH TO INTEGER
      CALL BCDREL(VALUE,PHRS)
      N=VALUE
      GO TO 200
160   WRITE(6,161)PINDEX,TYPE
161   FORMAT(/22H *** WARNING ***  IN  ,A8,2X,A6,
     1*ISN T A RECOGNIZED NAME LIST TYPE.  NAME LIST WILL NOT BE LOADED*
     1)
      LOAD=0
      GO TO 150
180   WRITE(6,181)PHRS
181   FORMAT(/16H *** WARNING ***,A10,
     1*ISN T A VALID NUMBER OF NAMES FOR NAME LIST  *
     2*NAME LIST WILL NOT BE LOADED*)
      GO TO 100
200   N=N+1
      IF(N.LE.1) GO TO 220
      IF(TYPE.EQ.TYPES(3))GO TO 210
C --->       READ NAMES FROM TAPE3
      READ(3,101)(NAMES(I),I=2,N)
      GO TO 220
```

```
C --->      READ TABLE NAMES
210   DO 215 I=2,N
      READ(3,211)NAMES(I),DIM
211   FORMAT(A3,G7.0)
      IDIM=DIM
      CALL PUTCOD(5,NAMES(I),IDIM)
215   CONTINUE
220   IF(N.LT.1)N=1
      NAMES(1)=N
C --->      WRITE NAMES ON MASS STORAGE PERMANENT FILE
      IF(LOAD.EQ.1)CALL WRITMS(78,NAMES,N,PINDEX)
      GO TO 100
C --->      COMMAND INTERPRETATION
300   GO TO(310,320,400,320,700,750),NTASK
C ====================  LIST STANDARD COMPONENTS === NTASK =1
310   LIST=1
      GO TO 100
C ====================  PURGE  NTASK = 2  OR  SYMBOL  ==  NTASK = 4
320   IF(ICPMAX.LT.0)GO TO 400
C --->      GET COMPONENT NAME
330   CALL NXTPH(ICOM,INDEX,COMNAM)
      IF(COMNAM.EQ.10H          )GO TO 100
C --->      LOCATE NAME IN COMPONENT NAME LIST
      DO 336 NCOMP=1,ICPMAX
      IF(KOMSTR(CMPNTS(NCOMP),1,2,COMNAM,1).EQ.0)GO TO 338
336   CONTINUE
      NCOMP=0
      GO TO 360
338   IF(NTASK.NE.2)GO TO 350
C --->      MOVE COMPONENT NAMES OVER ONE TO OVERWRITE PURGED NAME
      DO 340 I=NCOMP,ICPMAX
340   CMPNTS(I)=CMPNTS(I+1)
C --->      REDUCE NO. OF COMPONENTS
      ICPMAX=ICPMAX-1
      GO TO 330
350   CALL NXTPH(ICOM,INDEX,SYMB)
      CALL BCDREL(SYMB,SYMB)
      ISYMB=SYMB
      CALL PUTCOD(5,CMPNTS(NCOMP),ISYMB)
      ICPMOD=ICPMOD+1
      ICMMOD(ICPMOD)=NCOMP
      GO TO 330
360   WRITE(6,361)COMNAM
361   FORMAT(/33H *** WARNING ***  CAN T IDENTIFY ,A4,
     1*AS A STANDARD COMPONENT*)
      GO TO 330
C --->      GET COMPONENT NAME LIST FROM FILE 78
400   CALL READMS(78,ICPMAX,1,6HCMPNTS)
      CALL READMS(78,CMPNTS,ICPMAX,6HCMPNTS)
C --->      SHIFT NAMES OVER 1 WORD TO ELLIMINATE NO. OF WORDS
      DO 420 I=2,ICPMAX
420   CMPNTS(I-1)=CMPNTS(I)
      ICPMAX=ICPMAX-1
      IF(NTASK.LE.0)GO TO 130
      GO TO(130,330,720,330,130),NTASK
```

```
C --->      LIST COMPONENTS MODIFIED IF LIST=1
500   MAXCOM=ICPMOD
      IF(LIST.NE.1)GO TO 600
C --->      IF NO COMPS. MODIFIED, SKIP LISTING
      IF(MAXCOM.LE.0)GO TO 600
C --->      SCAN COMPONENTS SPECIFIED
      DO 560 I=1,MAXCOM
      J=I
      J=ICMMOD(I)
      COMNAM=CMPNTS(J)
520   CALL GETCOD(5,COMNAM,ISYMB)
      WRITE(6,521)I,COMNAM,ISYMB
521   FORMAT(//*COMPONENT NO.*,I3,*   NAME = *,A2,*   SYMBOL NO. = *,I3/
     1* INPUTS*,7X,*OUTPUTS*,6X,*TABLES*,7X,*DIMENSION*)
C --->      GET INPUT,OUTPUT,AND TABLE NAMES
      CALL COMDAT(COMNAM,4HINPT,NI,AINPUT)
      CALL COMDAT(COMNAM,4HOUTP,NO,OUTPUT)
      CALL COMDAT(COMNAM,4HTABS,NT,TABLE)
      MAX=MAXO(NI,NO,NT,1)
C --->      SCAN LONGEST LIST OF NAMES
      DO 550 J=1,MAX
C --->      BLANK NAMES
      AIN=10H
      OUT=10H
      TAB=10H
      ID=10H
      IF(J.LE.NI)AIN=AINPUT(J)
      IF(J.LE.NO)OUT=OUTPUT(J)
      IF(J.GT.NT)GO TO 540
      TAB=TABLE(J)
C --->      GET TABLE DIMENSION
      CALL GETCOD(5,TAB,ID)
540   WRITE(6,541)AIN,OUT,TAB,ID
541   FORMAT(2X,A10,3X,A10,3X,A8,5X,I4)
550   CONTINUE
560   CONTINUE
C --->      DEGAS MASS STORAGE FILE
C --->      IF NO COMPONENTS EXIST, CAUSE ABEND TO PREVENT DEGASSING
600   AIN=-1.
      IF(ICPMAX.LE.0)I=SQRT(AIN)
C ---       SORT COMPONENTS INTO ALPHABETICAL ORDER
      CALL CSORT(CMPNTS,ICPMAX)
C --->      SCAN ALL COMPONENTS
      DO 640 I=1,ICPMAX
C --->      LOAD COMPONENT NAME
      PINDEX=10H
      CALL STRMOV(CMPNTS(I),1,2,PINDEX,1)
C --->      SCAN THREE TYPES OF LISTS REQ D FOR EACH COMPONENT
      DO 640 J=1,3
      CALL STRMOV(TYPES(J),1,4,PINDEX,3)
C --->      READ LISTS FROM FILE 78
      CALL READMS(78,MAX,1,PINDEX)
      CALL READMS(78,AINPUT,MAX,PINDEX)
C --->      WRITE LISTS ONTO FILE 79
      CALL WRITMS(79,AINPUT,MAX,PINDEX)
640   CONTINUE
```

```
C --->      SHIFT COMPONENT NAMES OVER 1 WORD
      J=ICPMAX
      DO 660 I=1,ICPMAX
      CMPNTS(J+1)=CMPNTS(J)
660   J=J-1
C --->      ADD NO. OF COMPONENTS + 1 AS FIRST WORD IN LIST
      ICMP1=ICPMAX+1
C --->      STORE COMPONENT NAME LIST
      CALL WRITMS(79,CMPNTS,ICMP1,6HCMPNTS)
C ---       STORE PFNAME
      CALL READMS(78,PFNAME,1,6HPFNAME)
      CALL WRITMS(79,PFNAME,1,6HPFNAME)
      STOP
C =========  NEW FILE  === NTASK = 5
700   ICPMAX=0
      GO TO 100
C ==========  DUMP FILE  === NTASK = 3
720   CALL DUMPPF(CMPNTS,ICPMAX,MSI,TYPES,AINPUT)
      GO TO 100
C ==========  FILE NAME  === NTASK = 6
750   CALL NXTPH(ICOM,INDEX,PFNAME)
      CALL WRITMS(78,PFNAME,1,6HPFNAME)
      GO TO 100
      END
```

```
CGETCOD
      SUBROUTINE GETCOD(N,IARRAY,ICODE)
C  PURPOSE:  RETRIEVE A 4 DIGIT CODE,  VALUE OF CODE MUST BE BETWEEN -
C            2047 , STORED 5 CODES/WORD, FROM AN ARRAY OF PARAMETER
C            CODES.  THIS ROUTINE IS USED TO REDUCE THE STORAGE REQUIRED
C            TO STORE THE I/O CODE LISTS FOR EACH ANALYSIS MODULE.
C  CALL SEQUENCE:  N   LOCATION OF CODE IN ARRAY IARRAY  5 CODES/WORD .
C                  IARRAY   INTEGER ARRAY WHICH RECIEVES CODE NUMBER.
C                  ICODE   VALUE OF CODE INPUT TO ROUTINE.
      DIMENSION IARRAY(1)
      DATA MASK/7777B/
C  DETERMINE WHICH WORD IN ARRAY CONTAINS THE NTH CODE.
      IWORD=(N-1)/5+1
C  DETERMINE THE NUMBER OF BITS TO SHIFT CODE TO RIGHT MOST 12 BITS.
      ISHIFT=(MOD(N-1,5)-4)*12
C  SHIFT CODE BITS TO RIGHT HAND POSITION.
      ICODE=SHIFT(IARRAY(IWORD),ISHIFT)
C  MASK OUT UNWANTED BITS TO LEFT OF CODE.
      ICODE=MASK.AND.ICODE
C  TEST SIGN BIT.
      IF(ICODE.LT.2048) RETURN
C  RESTORE 1 BITS FOR NEGATIVE CODE.
      ICODE=ICODE.OR..N.MASK
      RETURN
      END
```

```
GETT
          IDENT  GETT
          ENTRY  GETT
 +        VFD 18/0HGET,42/3
 GET      BSSZ 1
 GETT     EQU    GET
          SB4  EQ-1          . INITIALIZE MULTIPLE OF 10 COUNTER.
          SA4    A1-B4       .
          SA2    X4          . PUT I IN X2.
 LOOP     SB4  B4+1          . COUNT MULTIPLES OF 10.
          SX2  X2-10         . SUBTRACT 10 FROM I.
          ZR   X2,OK         . IF X2 .LE. 0, EXIT FROM LOOP.
          PL   X2,LOOP       . LOOP UNTIL NO MORE MULTIPLES OF 10 IN I.
 OK       SA4    X1+B4       . LOAD WORD CONTAINING ITH CHAR. OF S.
          MX7    6           .
          SA5  SIX           .
          SX2  X2+9          . X2 = POSITION IN S WORD.
          PX2  B0,X2         . PACK X2
          DX2  X2*X5         . MULTIPLY BY SIX
          SB2  X2            .
          LX4    B2,X4       . LEFT ADJUST ITH CHARACTER.
          SA5  MASK2         .
          BX6    X4*X7       . MASK OUT REMAINDER OF WORD.
          BX6  X5+X6         . FILL X6 WITH BLANKS.
          SA4    A1+2        .
          SA6    X4          . STORE IN T.
          EQ   B0,B0,GET
 MASK2    DATA 0055555555555555555B
 SIX      DATA 2000000000000000006B
          END
```

```
ISCAN
          IDENT  ISCAN
*
*         WRITTEN BY TOM BAKER, EDT SUPPORT   PHONE 655-6509     12/1/70
*
*
*         THIS IS A COMPLETE REWRITE OF THE OLD SYSTEM ROUTINE.
*           DIFFERENCES - 9 TO 15 TIMES FASTER
*                       - CALLS NO SUBPROGRAMS (USED TO CALL KOMSTR)
*                       - HAS A SPECIAL LOOP FOR THE CASE WHERE THE
*                           STRING 2 SET IS ONLY ONE CHARACTER
*
* CALLING SEQUENCE   J2=ISCAN(S1,I1,N1,S2,I2,N2,J1)
*         S1 - STARTING ADDRESS OF THE FIRST STRING
*         I1 - CHARACTER POSITION IN S1 OF FIRST CHARACTER TO BE COMPARED
*         N1 - THE NUMBER OF CHARACTERS IN S1 TO BE COMPARED.  IF N1
*              IS NEGATIVE, THE SCAN IS RIGHT-TO-LEFT IN S1.
*         S2 - STARTING ADDRESS OF THE SECOND STRING
*         I2 - CHARACTER POSITION IN S2 OF FIRST CHARACTER TO BE COMPARED
*         N2 - THE NUMBER OF CHARACTERS IN S2 TO BE COMPARED.
*         J1 - OUTPUT, CHARACTER POSITION IN S1 WHERE A MATCH WAS FOUND
*         J2 - OUTPUT, CHARACTER POSITION IN S2 WHERE A MATCH WAS FOUND
*         J1=J2=0   NO MATCH WAS FOUND
*         J1=J2=-1  INPUT ERROR
*
*
          ENTRY  ISCAN
          VFD    30/0LISCAN,30/7
 ISCAN    BSSZ   1
          MX0    52B
          SX6    A0                 SAVE A0 OF CALLING PROGRAM
          SA6    SAVEA0
          SA0    A1                 SET UP A0 FOR USE BY ISCAN
          SA2    A0+1               FETCH THE FIRST STRING STARTING
          SA2    X2                   CHARACTER POINTER (I1)
          SA4    A0+4               FETCH THE SECOND STRING STARTING
          SA4    X4                   CHARACTER POINTER (I2)
          ZR     X2,ERROR           I1 MUST BE A NON-ZERO
          NG     X2,ERROR             POSITIVE NUMBER
          SA5    A0+2
          SA2    A0+5
          SA5    X5                 GET THE VALUE OF N1
          SA2    X2                 GET THE VALUE OF N2
          ZR     X4,ERROR           I2 MUST BE A NON-ZERO
          NG     X4,ERROR             POSITIVE NUMBER
          ZR     X5,ERROR           N1 MUST NOT EQUAL ZERO
          ZR     X2,ERROR           N2 MUST BE A NON-ZERO
          NG     X2,ERROR             POSITIVE NUMBER
* A LEGITIMATE REQUEST HAS BEEN MADE - PROCEED
          SB1    X5                 B1 CONTAINS THE VALUE OF N1
          SB4    X2                 B4 CONTAINS THE VALUE OF N2
          MX1    73B                X1 CONTAINS A MINUS ONE
          AX5    77B                FILL X5 WITH THE SIGN BIT OF N1
          BX6    -X1-X5             X6 CONTAINS A ONE WITH THE SIGN OF N1
          SB6    X6                 B6 CONTAINS A ONE WITH THE SIGN OF N1
          SB7    6                  B7 WILL BE USED TO CONTROL THE LEFT
          PL     X6,S1                SHIFT OF STRING ONE - 6 BITS AT A
```

```
            SB7     54                  TIME IF N1 IS +VE, 54 BITS IF -VE
 S1         SA3     TEN                 A PACKED UNNORMALIZED DECIMAL TEN
            SA2     TENTH               A FLOATING POINT TENTH
            SB3     X1                  B3 CONTAINS A MINUS ONE
            SX4     X4+B3
            PX4     B0,X4               I2-1 AS A PACKED UNNORMALIZED NUMBER
            NX6     B5,X4
            FX6     X6*X2               NUMBER OF WORDS STARTING POINT IS
*                                         PAST THE LOCATION OF S2
            UX6     B5,X6
            SA5     A0+3
            LX6     B5,X6
            IX7     X5+X6               CALCULATE STARTING ADDRESS OF THE
            SA7     W1S2                  SECOND STRING AND STORE IN W1S2
            PX6     B0,X6               X6 CONTAINS THE NUMBER OF WORDS THAT
*                                         THE STARTING POINT IS PAST THE
*                                         LOCATION OF S2 AS A PACKED
*                                         UNNORMALIZED NUMBER
            DX6     X6*X3
            UX6     B5,X6               CALCULATE THE
            IX6     X4-X6                 CHARACTER POSITION MINUS ONE
            SA5     SIX                   OF THE FIRST CHARACTER IN S2,
            DX6     X6*X5                 MULTIPLY IT BY SIX, AND STORE THE
            UX6     B5,X6                 RESULTING BIT SHIFT COUNT
            SA6     W1S2P                 IN W1S2P
            SA4     A0+1                FETCH THE FIRST STRING STARTING
            SA4     X4                    CHARACTER POINTER (I1)
            SX4     X4+B3
            PX6     B0,X4               I1-1 AS A PACKED UNNORMALIZED NUMBER
            NX6     B5,X6
            SA1     A0                  FETCH LOCATION OF S1
            FX6     X6*X2               NUMBER OF WORDS STARTING POINT IS
*                                         PAST THE LOCATION OF S1
            UX6     B5,X6
            LX6     B5,X6
            SB5     X1
            SA1     X6+B5               X1 CONTAINS THE FIRST WORD IN S1
*                                         TO BE CHECKED
            PX6     B0,X6               X6 CONTAINS THE NUMBER OF WORDS THAT
*                                         THE STARTING POINT IS PAST THE
*                                         LOCATION OF S1 AS A PACKED
*                                         UNNORMALIZED NUMBER
            DX6     X6*X3
            UX6     B5,X6               CALCULATE THE
            IX6     X4-X6                 CHARACTER POSITION MINUS ONE
            PX7     B0,X6                 OF THE FIRST CHARACTER IN S1
            SB2     B2-B2
            LT      B6,B0,SM
            DX5     X7*X5               LEFT TO RIGHT SCAN OF S1 WAS
            IX7     X3-X7                 REQUESTED, CALCULATE INITIAL VALUE
*                                         OF S1 POSITION IN WORD INDEX
            SB5     X5                    AND SET UP B5 FOR INITIAL
            EQ      S                     SHIFTING OF FIRST S1 WORD
 SM         MX4     73B                 RIGHT TO LEFT SCAN OF S1 WAS
            DX5     X7*X5                 REQUESTED, CALCULATE INITIAL VALUE
*                                         OF S1 POSITION IN WORD INDEX
```

```
          UX5     B5,X5               AND SET UP B5 FOR INITIAL
          SB5     X5-48                 SHIFTING OF FIRST S1 WORD
          IX7     X6-X4
          GE      B5,B0,S
          SB5     B5+60
 S        MX0     66B               SET UP CHARACTER MASK
          LX1     B5,X1
          MX6     73B               X6 CONTAINS A MINUS ONE
          SB3     B3+B4             IF STRING 2 IS ONE CHARACTER LONG,
          EQ      B3,B0,S21C          GO TO SPECIAL LOOP
          SX5     10
 SMX1     LX1     B7,X1             FETCH NEXT (OR FIRST) CHARACTER FROM
          SB2     B2+B6               S1, INCREMENT S1 CHARACTER POSITION
          IX7     X7+X6               INDEX, DECREMENT S1 POSITION IN
          BX3     -X0*X1              WORD INDEX, ISOLATE NEXT CHARACTER
*                                     IN S1
          SA2     W1S2              FETCH S2 POINTERS
          SA4     W1S2P
          SA2     X2
          SB3     X4
          LX2     B3,X2             PREPARE TO EXTRACT FIRST CHARACTER
          SB5     B5-B5               FROM S2 STRING
          SB3     B3-60
 SMX2     LX2     6                 SHIFT NEXT CHARACTER IN S2 STRING TO
*                                     RIGHTMOST POSITION IN X2,
          SB5     B5+1                INCREMENT S2 CHARACTER INDEX,
          BX4     -X0*X2              EXTRACT NEXT CHARACTER FROM S2
          SB3     B3+6                INCREMENT S2 POSITION IN WORD INDEX
          BX4     X3-X4               AND COMPARE THE TWO CHARACTERS
          ZR      X4,MATCH          IF THE CHARACTERS MATCH, GO TO MATCH
          EQ      B4,B5,ES2S        IF THE END OF THE S2 STRING HAS BEEN
*                                     REACHED, GO TO ES2S
          LT      B3,B0,SMX2        IF THERE ARE MORE CHARACTERS IN THE
*                                     CURRENT S2 WORD TO BE COMPARED,
*                                     GO BACK TO SMX2 AND WORK ON THE
*                                     NEXT CHARACTER IN S2
          SA2     A2+1              OTHERWISE, GET THE NEXT WORD IN S2
          SB3     -60                 INITIALIZE THE POSITION IN WORD
          EQ      SMX2                INDEX AND GET BACK TO WORK ON S2
 ES2S     EQ      B2,B1,NM          IF ALL THE CHARACTERS IN S1 HAVE
*                                     BEEN CHECKED, GO TO NM (NO MATCH)
*                                     OTHERWISE, CHECK TO SEE IF THE
*                                     CURRENT WORD IN X1 HAS ANYMORE
          NZ      X7,SMX1             CHARACTERS LEFT TO CHECK. IF SO,
*                                     GO GET THE NEXT CHARACTER. IF NOT,
          SA1     A1+B6               GET THE NEXT WORD IN S1, INITIALIZE
          BX7     X5                  THE POSITION IN WORD INDEX AND
          LT      B0,B6,SMX1          RETURN. NOTE THAT IF THE SCAN IS
          LX1     6                   RIGHT TO LEFT, AN EXTRA SHIFT IS
          EQ      SMX1                REQUIRED WHICH NEUTRALIZES THE
*                                     SHIFT AT SMX1.
 S21C     SA2     W1S2              SPECIAL ONE CHARACTER STRING 2 LOOP
          SA4     W1S2P             FETCH THE S2 POINTERS
          SA2     X2
          SB3     X4+6
          LX2     B3,X2
          SB5     B4
```

```
            BX4     -X0*X2              EXTRACT THE S2 CHARACTER
            SX5     10
 SMX11C     LX1     B7,X1               FETCH NEXT (OR FIRST) CHARACTER FROM
            SB2     B2+B6                 S1, INCREMENT S1 CHARACTER POSITION
            IX7     X7+X6                 INDEX, DECREMENT S1 POSITION IN
            BX3     -X0*X1                WORD INDEX, ISOLATE THE NEXT
            BX3     X3-X4                 CHARACTER IN S1, COMPARE IT WITH
            ZR      X3,MATCH              THE S2 CHARACTER AND IF THE
*                                         CHARACTERS MATCH, GO TO MATCH
            EQ      B2,B1,NM            IF ALL THE CHARACTERS IN S1 HAVE
*                                         BEEN CHECKED, GO TO NM (NO MATCH)
*                                         OTHERWISE, CHECK TO SEE IF THE
*                                         CURRENT WORD IN X1 HAS ANYMORE
            NZ      X7,SMX11C             CHARACTERS LEFT TO CHECK. IF SO,
*                                         GO GET THE NEXT CHARACTER. IF NOT,
            SA1     A1+B6                 GET THE NEXT WORD IN S1, INITIALIZE
            BX7     X5                    THE POSITION IN WORD INDEX AND
            LT      B0,B6,SMX11C          RETURN. NOTE THAT IF THE SCAN IS
            LX1     6                     RIGHT TO LEFT, AN EXTRA SHIFT IS
            EQ      SMX11C                REQUIRED WHICH NEUTRALIZES THE
*                                         SHIFT AT SMX11C.
 NM         SA1     A0+6                NO MATCH WAS FOUND
            SX6     B5-B5                 SET J1 AND J2 EQUAL TO ZERO
            SA6     X1                    AND RETURN
            SA1     SAVEA0              RESTORE THE CALLING PROGRAMS
            SA0     X1                    A0 REGISTER
            EQ      B4,B4,ISCAN
 MATCH      SA2     A0+1                A MATCH HAS BEEN FOUND.
            SA3     A0+4                  J1 IS RETURNED AS THE POSITION IN
*                                         S1 OF THE MATCH.  (J1=I1+B2-B6)
            SB3     B2-B6                 J2 IS RETURNED AS THE POSITION IN
            SA1     A0+6                  S2 OF THE MATCH.  (J2=I2+B5-1)
            SA2     X2
            SB4     B5-1
            SA3     X3
            SX7     X2+B3
            SX6     X3+B4
            SA7     X1
            SA1     SAVEA0              RESTORE THE CALLING PROGRAMS
            SA0     X1                    A0 REGISTER
            EQ      B1,B1,ISCAN
 ERROR      SA1     A0+6                AN ERROR HAS BEEN FOUND IN THE INPUT
            MX6     73B                   REQUEST.  J1 AND J2 ARE RETURNED
            SA6     X1                    AS MINUS ONE.
            SA1     SAVEA0              RESTORE THE CALLING PROGRAMS
            SA0     X1                    A0 REGISTER
            EQ      B1,B1,ISCAN
 SAVEA0     BSSZ    1                   STORAGE FOR CALLING PROGRAMS A0
 W1S2       BSS     1                   ADDRESS OF WORD CONTAINING FIRST
*                                         CHARACTER IN S2 TO BE CHECKED
 W1S2P      BSS     1                   POINTER TO CONTROL SHIFT OF FIRST
*                                         WORD IN S2 TO PROPERLY POSITION
*                                         THE FIRST CHARACTER FOR CHECKING
 TENTH      DATA    .1                     A FLOATING POINT TENTH
 TEN        DATA    0200000000000000000012  A PACKED UNNORMALIZED TEN
 SIX        DATA    0200000000000000000006  A PACKED UNNORMALIZED SIX
      END
```

```
KOMPAR
          IDENT  KOMPAR      (S,T,I)
***
*
*      SUBJECT:   COMPASS SUBROUTINE KOMPAR
*
*      AUTHORS:   ANDERSON, O. L.   -   LA ROWE, E.
*                 BCS 6600 METHODS GROUP - 22 FEBRUARY 1971
*
*      PURPOSE:   TO PERFORM A 60 BIT LOGICAL COMPARE OF TWO WORDS
*
*      USAGE:     CALL KOMPAR (S,T,I)
*                 I =  KOMPAR (S,T,I)
*
*                 WHERE
*      INPUT
*
*                 S      = FIRST STRING WORD
*                 T      = SECOND STRING WORD
*
*      OUTPUT
*
*                 I      = -1 IF S < T
*                        =  0 IF S = T
*                        =  1 IF S > T
*
          ENTRY  KOMPAR
*
          VFD    36/0LKOMPAR,24/3
*
 KOMPAR   BSSZ   1             ENTRY/EXIT - KOMPAR
          SA5    X1            GET S
          MX7    1
          SA2    A1+1
          SA2    X2            GET T
          LX5    59
          BX3    -X7*X5
          LX2    59
          BX4    -X7*X2
          MX6    59
          SA1    A1+2
          IX7    X3-X4         LEFT PARTIAL DIFFERENCE
          NG     X7,KOMPAR1    IF S < T
          BX6    -X6
          NZ     X7,KOMPAR1    IF S > T
*
*                LEFTMOST 59 BITS ARE EQUAL.  TEST RIGHTMOST ONES.
*
          IX6    X5-X2
          LX6    1
 KOMPAR1  SA6    X1            STORE I
          EQ     KOMPAR        RETURN
*
*
*
          END
```

```
KOMSTR
          IDENT  KOMSTR      .CALL KOMSTR(SA,LA,NA,SB,LB)
**
*      KOMSTR   REVISED TO ECONOMIZE ON FIELD LENGTH, EXECUTION TIME.
*               REVISED BY   BRUCE BAILEY
*                        BCS SCIENTIFIC SYSTEMS
**        COMPARES CHARS LA THRU LA+NA-1 OF SA TO CHARS LB THRU LB+NA-1
**        OF SB ALPHANUMERICALLY.  RETURNS - 0 + AS SA LT EQ GT SB.
**
*
          ENTRY     KOMSTR
 +        VFD       36/0HKOMSTR,24/5
 KOMSTR   PS
          SB1       1
          SA4       A1+B1
          SA2       A4+B1
          SA4       X4              LA
          SA2       X2              NA
          SA3       A1+4
          MX6       0               INIT VALUE
          SA3       X3              LB
          ZR        X4,KOMSTR
          NG        X4,KOMSTR
          ZR        X2,KOMSTR
          NG        X2,KOMSTR
          ZR        X3,KOMSTR
          NG        X3,KOMSTR
**
**        INPUT OK - START LOADING REGISTERS.
**
          SB7       B0+11
          SB2       X4              LA
          SB6       B7-B1
          LT        B2,B7,V20
 V10      SX1       X1+B1           ADR OF FIRST SA WORD
          SB2       B2-B6           CHAR COUNT (SAC)
          GE        B2,B7,V10
 V20      SB3       B2+B2           2*SAC
          SA4       A1+3
          SB2       B3+B3           4*SAC
          SA1       X1              INIT VALUE (X1 AND A1)
          SB3       B2+B3           6*SAC
          SB2       B3-60           6*SAC-60
          SB5       X3              LB
          SB2       B0-B2           60-6*SAC, THE INIT VALUE.
          LT        B5,B7,V40
 V30      SX4       X4+B1           ADR OF FIRST SB WORD
          SB5       B5-B6           CHAR COUNT (SBC)
          GE        B5,B7,V30
 V40      SB7       B5+B5           2*SBC
          SB5       B7+B7           4*SBC
          MX0       54              INIT VALUE
          SB7       B5+B7           6*SBC
          SA4       X4              INIT VALUE (X4 AND A4)
          SB5       B7-60           6*SBC-60
```

```
          SB3       X2              INIT VALUE (NA)
          SB5       B0-B5           60-6*SBC, THE INIT VALUE.
          SB7       B0+6            INIT VALUE
          SB4       B0+54           INIT VALUE
**
**        IN-STACK EXECUTION LOOP.
**
 V50      AX2       X1,B2           SA WORD
          BX2       -X0*X2          SA CHAR
          SB3       B3-B1
          AX5       X4,B5           SB WORD
          BX5       -X0*X5          SB CHAR
          IX7     X2-X5
          SB2       B2-B7           SA SHIFT COUNTER.
          NZ      X7,ALMOST
**                                  INCREMENT COUNTERS.
          EQ        B3,B0,KOMSTR    IF EQUAL STRINGS.
          GE        B2,B0,V60       IF RESET NOT YET NECESSARY.
          SA1       A1+B1           NEXT SA WORD, ALSO RESET A1.
          SB2       B4              RESET COUNTER    .
 V60      SB5       B5-B7           SB SHIFT COUNTER
          GE        B5,B0,V50       IF RESET NOT YET NECESSARY.
          SA4       A4+B1           NEXT SB WORD, ALSO RESET A4.
          SB5       B4              RESET COUNTER
          EQ        V50             KEEP LOOKING.
 ALMOST   SX6     B1
          PL      X7,KOMSTR
          BX6     -X6
          EQ      KOMSTR
          END
```

```
CLCMPH
      SUBROUTINE LCMPH(IPHRS,ICOML,ICLMAX,ICLMIN,LOC)
C  PURPOSE:  LOCATE PHRASE IN STRING OF COMMAND PHRASES
C  CALL SEQUENCE:  IPHRS  - PHRASE TO BE IDENTIFIED
C                  ICOML  - LIST OF COMMAND PHRASES
C                  ICLMAX - MAX. NO. OF COMMAND PHRASES TO SEARCH
C                  ICLMIN - MIN. NO. OF COMMAND PHRASES TO SEARCH
C                  LOC    - LOCATION OF IPHRS IN ICOML
C                    (LOC = 0  IF PHRASE NOT FOUND)
C  DESIGNED BY:  J.D. BURROUGHS                    OCT 1973
      DIMENSION ICOML(ICLMAX)
      IF(ICLMIN.LT.1)ICLMIN=1
      IF(ICLMAX.LT.ICLMIN)ICLMAX=ICLMIN
C ==========  ASSURE THAT SEARCH STARTS BETWEEN ICLMIN AND ICLMAX
      IF(LOC.LT.ICLMIN.OR.LOC.GT.ICLMAX)LOC=ICLMIN
C ========  SAVE STARTING POINT OF SEARCH
      LOCS=LOC
100   IF(IPHRS.NE.ICOML(LOC)) GO TO 300
      RETURN
300   LOC=LOC+1
C =======  RETURN TO START IF LAST COMMAND PHRASE IS REACHED
      IF(LOC.GT.ICLMAX) LOC=ICLMIN
C ==========  STOP SEARCH WHEN STARTING POINT IS REACHED
      IF(LOC.NE.LOCS) GO TO 100
      LOC=0
      RETURN
      END
```

```
CNUMERC
      SUBROUTINE NUMERC(PHRS),RETURNS(A)
C  PURPOSE:  TO DETECT WHEN THE LEFT MOST CHARACTER IN A STRING
C            IS NUMERIC
C  CALL SEQUENCE:  PHRS - STRING OF CHARACTERS
C            RETURNS(A) - RETURN TAKEN IF CHARACTER IS NOT NUMERIC
      DIMENSION NUM(2)
      DATA NUM/20H1234567890-.+       /
C --->      COMPARE FIRST CHARACTER TO NUMERICS
      I=ISCAN(PHRS,1,1,NUM,1,14,M1)
      IF(I.LE.0) RETURN A
      RETURN
      END
```

```
CNXTPH
      SUBROUTINE NXTPH(ICOM,INDEX,IPHRS)
C  PURPOSE:  LOCATE NEXT PHRASE IN COMMAND STRING.
C  CALL SEQUENCE: ICOM  - COMMAND STRING
C                 INDEX - INDEX TO NEXT CHARACTER TO BE EXAMINED
C                 IPHRS - NEXT PHRASE (RETURNED BLANK IF NONE FOUND)
C  DELIMITERS:  3 OR MORE BLANKS, COMMA, EQUALS, LEFT OR RIGHT PARENTHES
      COMMON/CIO/IREAD,IWRITE,IDIAG
      DIMENSION ICOM(1)
      DATA IBLNK/10H          /
      IPMAXC=10
      ICMAXC=80
      IPHRS=IBLNK
C ---       RETURN IF AT COLUMN 80
      IF(INDEX.GE.ICMAXC)RETURN
C ---       LOCATE FIRST NON-BLANK, NON-DELIMITER CHARACTER
150   DO 200 I=INDEX,ICMAXC
      CALL GETT(ICOM,I,KAR)
      IF(KAR.EQ.1H,.OR.KAR.EQ.1H=.OR.KAR.EQ.1H(.OR.KAR.EQ.1H))GO TO 200
      IF(KAR.NE.IBLNK) GO TO 300
200   CONTINUE
      INDEX=ICMAXC
      IF(IDIAG.GE.100.)WRITE(IWRITE,251)INDEX,IPHRS
251   FORMAT(14HNXTPHR2 INDEX=,I3,1X,A10)
C ---       RETURN WHEN REST OF STRING IS EMPTY
      RETURN
C ---       LOCATE NEXT DELIMITER  (END OF PHRASE)
300   ISTART=I
      DO 400 I=ISTART,ICMAXC
      CALL GETT(ICOM,I,KAR)
      IF(KAR.EQ.1H,.OR.KAR.EQ.1H=.OR.KAR.EQ.1H(.OR.KAR.EQ.1H))GO TO 490
      IF(KAR.EQ.IBLNK) GO TO 350
      INBLNK=0
      GO TO 400
350   IF(INBLNK.GE.2) GO TO 500
      INBLNK=INBLNK+1
400   CONTINUE
      INDEX=ICMAXC
      GO TO 600
490   ISTOP=I-1
      GO TO 510
500   ISTOP=I-3
510   INDEX=I
C ---       TEST TO LIMIT PHRASE TO <= 10 CHARACTERS
      IF(ISTOP-ISTART+1.LE.IPMAXC) GO TO 700
600   ISTOP=ISTART+IPMAXC-1
C ---       TEST TO PREVENT PHRASE FROM GOING BEYOUND COL. 80
      IF(ISTOP.GT.ICMAXC) ISTOP=ICMAXC
700   INBLNK=ISTOP-ISTART+1
C ---       LOAD PHRASE
      CALL STRMOV(ICOM,ISTART,INBLNK,IPHRS,1)
      IF(IDIAG.GE.100.)WRITE(IWRITE,801)INDEX,IPHRS
801   FORMAT(13HNXTPHR INDEX=,I3,1X,A10)
      RETURN
      END
```

```
CPUTCOD
      SUBROUTINE PUTCOD(N,IARRAY,ICODE)
C  PURPOSE:  PLACE A 4 DIGIT CODE,  VALUE OF CODE MUST BE BETWEEN -
C            2047 , STORED 5 CODES/WORD, FROM AN ARRAY OF PARAMETER
C            CODES.  THIS ROUTINE IS USED TO REDUCE THE STORAGE REQUIRED
C            TO STORE THE I/O CODE LISTS FOR EACH ANALYSIS MODULE.
C  CALL SEQUENCE:  N   LOCATION OF CODE IN ARRAY IARRAY  5 CODES/WORD .
C                  IARRAY   INTEGER ARRAY WHICH RECIEVES CODE NUMBER.
C                  ICODE   VALUE OF CODE INPUT TO ROUTINE.
      DIMENSION IARRAY(1)
      MASK=7777B
C  DETERMINE WHICH WORD IN ARRAY IS TO BE MODIFIED.
      IWORD=(N-1)/5+1
C  DETERMINE NO. OF BITS TO SHIFT CODE TO LEFT.
      ISHIFT=(4-MOD(N-1,5))*12
C  SHIFT CODE + MASK TO PROPER BIT LOCATION IN WORD.
      ICOD=SHIFT(ICODE,ISHIFT)
      MASK=SHIFT(MASK,ISHIFT)
C  PLACE CODE BITS INTO CORRECT LOCATION IN WORD OF IARRAY.
      IARRAY(IWORD)=(IARRAY(IWORD).AND..N.MASK).OR.(ICOD.AND.MASK)
      RETURN
      END
```

```
PUTT
          IDENT   PUTT
          ENTRY  PUTT
 +        VFD 18/0HPUT,42/3
 PUT      BSSZ 1
 PUTT     EQU    PUT
          SB4   B0-1          . INITIALIZE MULTIPLE OF 10 COUNTER.
          SA4     A1-B4       .
          SA2     X4          . PUT I IN X2.
 LOOP     SB4   B4+1          . COUNT MULTIPLES OF 10.
          SX2   X2-10         . SUBTRACT 10 FROM I.
          ZR    X2,OK         . IF X2 .LE. 0, EXIT FROM LOOP.
          PL    X2,LOOP       . LOOP UNTIL NO MORE MULTIPLES OF 10 IN I.
 OK       SA4     X1+B4       . LOAD WORD TO RECEIVE THE CHARACTER.
          MX7     6           .
          SA5   SIX           .
          SX2   X2-1          .
          PX2   B0,X2         . PACK X2
          DX2   X2*X5         . MULTIPLY BY SIX
          SA3     A1+2        .
          SA3     X3          . PUT T IN X3.
          SB2   X2            .
          BX3     X3*X7       . MASK OUT LAST 9 CHAR. OF T.
          AX3   B2,X3         . SHIFT CHARACTER INTO POSITION.
          AX7     B2,X7       . SHIFT MASK INTO POSITION.
          BX6     -X7*X4      . MASK OUT CHARACTER IN S.
          BX6   X3+X6         . OR IN CHARACTER FROM T.
          SA6     X1+B4       . STORE IN PROPER POSITION IN STRING S.
          EQ    B0,B0,PUT
 SIX      DATA 2000000000000000006B
          END
```

```
STRMOV
          IDENT  STRMOV       .CALL STRMOV(SA,LA,NA,SB,LB)
**
*      STRMOV   REVISED TO ECONOMIZE ON FIELD LENGTH, EXECUTION TIME.
*               REVISED BY  BRUCE BAILEY
*                        BCS SCIENTIFIC SYSTEMS
**        MOVES NA CHARS FROM SA, STARTING IN POSITION LA, INTO SB,
**        STARTING IN POSITION LB, A CHAR AT A TIME.  REQUIRES LA,NA,LB
**        ALL POSITIVE INTEGERS.  IF INPUT ERROR, NO ACTION TAKEN.
**
          ENTRY  STRMOV
**
**        STORE FINAL SB WORD - CAN BE A REDUNDANT STORE.
 V70      BX6       X4
          SA6       A4
          EQ      B0,B0,STRMOV
**
          VFD     36/0LSTRMOV,24/5
 STRMOV   PS
          SB1       1
          SA5       A1+B1
          SA2       A5+B1
          SA5       X5              LA
          SA2       X2              NA
          ZR        X5,STRMOV       CHECK
          NG        X5,STRMOV       LA AND
          SA3       A1+4            LB
          SA3       X3
          SB2       X5              LA
          ZR        X2,STRMOV       NA FOR
          NG        X2,STRMOV       0 OR NEG
          ZR        X3,STRMOV       CHECK
          NG        X3,STRMOV       LB
**
**        INPUT OK - START LOADING REGISTERS.
**
          SB7       B0+11           TEMP TO DECOMPOSE LA AND LB.
          SB6       B7-B1
          LT        B2,B7,V20       LA.LT.11
 V10      SX1       X1+B1           TEMP ADR OF SA, FIRST WORD.
          SB2       B2-B6           TEMP CHAR COUNT, SA.  (ABBR SAC)
          GE        B2,B7,V10
 V20      SB3       B2+B2           B3=2*SAC
          SA4       A1+3
          SB2       B3+B3           B2=4*SAC
          SA1       X1              INIT VALUE (X1 AND A1)
          SB3       B2+B3           B3=6*SAC
          SB2       B3-60           B2=6*SAC-60
          SB5       X3              LB
          SB2       B0-B2           B2=60-6*SAC, THE INIT VALUE.
          LT        B5,B7,V40       LB.LT.11
 V30      SX4       X4+B1           TEMP ADR OF SB, FIRST WORD.
          SB5       B5-B6           TEMP CHAR COUNT, SB (ABBR SBC)
          GE        B5,B7,V30
 V40      SB6       B5+B5           2*SBC
          SB7       B6+B6           4*SBC
```

```
          SB5       B6+B7           6*SBC, THE INIT VALUE.
          SA4       X4              INIT VALUE (X4 AND A4)
          SB7       B5-60           6*SBC-60
          SB3       X2              INIT VALUE (NA)
          SB6       B0-B7           60-6*SBC, THE INIT VALUE.
          MX0       54              INIT VALUE
          SB7       B0+6            INIT VALUE
**
**        EXECUTION LOOP, IN-STACK.  FETCHES AND STORES AS NEEDED.
**
 V50      AX2       B2,X1           MOVE SA CHAR TO POS 10
          BX3       -X0*X2          CHAR
          LX5       B5,X4           POSITION SB WORD.
          SB2       B2-B7           DECREMENT SA SHIFT COUNTER.
          BX7       X0*X5           0 POSITION 10
          SB3       B3-B1           NB=NB-1
          BX6       X7+X3           SA CHAR IN SB WORD.
          LX4       B6,X6           REPOSITION SB WORD.
**                                  INCREMENT COUNTERS.
          EQ        B3,B0,V70       IF NB.EQ.0 (I.E., DONE)
          GE        B2,B0,V60       IF RESET NOT YET NECESSARY.
          SA1       A1+B1           RESET A1 AND X1 FOR NEXT SA WORD.
          SB2       B0+54           RESET B2 TO 54
 V60      SB5       B5+B7           INCREMENT FIRST SB SHIFT COUNTER.
          SB6       B6-B7           DECREMENT SECOND SB SHIFT COUNTER.
          GE        B6,B0,V50       IF RESET NOT YET NECESSARY.
          BX6       X4              REVISED SB WORD
          SA6       A4              STORE REVISED SB WORD
          SB5       B7              RESET B5 TO 6
          SA4       A4+B1           FETCH NEXT SB WORD, RESET A4
          SB6       B0+54           RESET B6 TO 54
          EQ        V50             RECYCLE
          END
```

5.0 PRINTER PLOT PROGRAM

Lineprinter plots of simulation results are produced by a postprocessor program NSMPPT. This program is executed after the completion of the simulation program. NSMPPT reads simulation and scaling data from file TAPE30 and produces the requested line printer plots. Figure 5.1-1 shows the macro flow diagram of NSMPPT.

Each unique channel of plot data is stored on file TAPE30. Channels, such as TIME, which may be used by several plots are stored only once. The format data describes how the channels are to be combined to form the plots. The individual channel data are loaded into an array DSPLY. The data for each plot is then scaled and transformed to hollerith form and placed in the array GRAPHR. Title and scale information are also placed in this array to form the final plot configuration.

The contents of GRAPHR are printed on the lineprinter to produce each plot.

5.1 PRINTER PLOT PROGRAM SOURCE LISTINGS

Compilation listings for the NSMPPT program follows. The names of the routines, listed in alphabetical order, are:

CENTER	NSMPPT
GNFPLT	PLOTC
GRIDLI	QPPLOT
LEFTT	QXMXMN
LINPLT	RTLPLT
MNMX	SIMPLT
NCHAR	TNFPLT

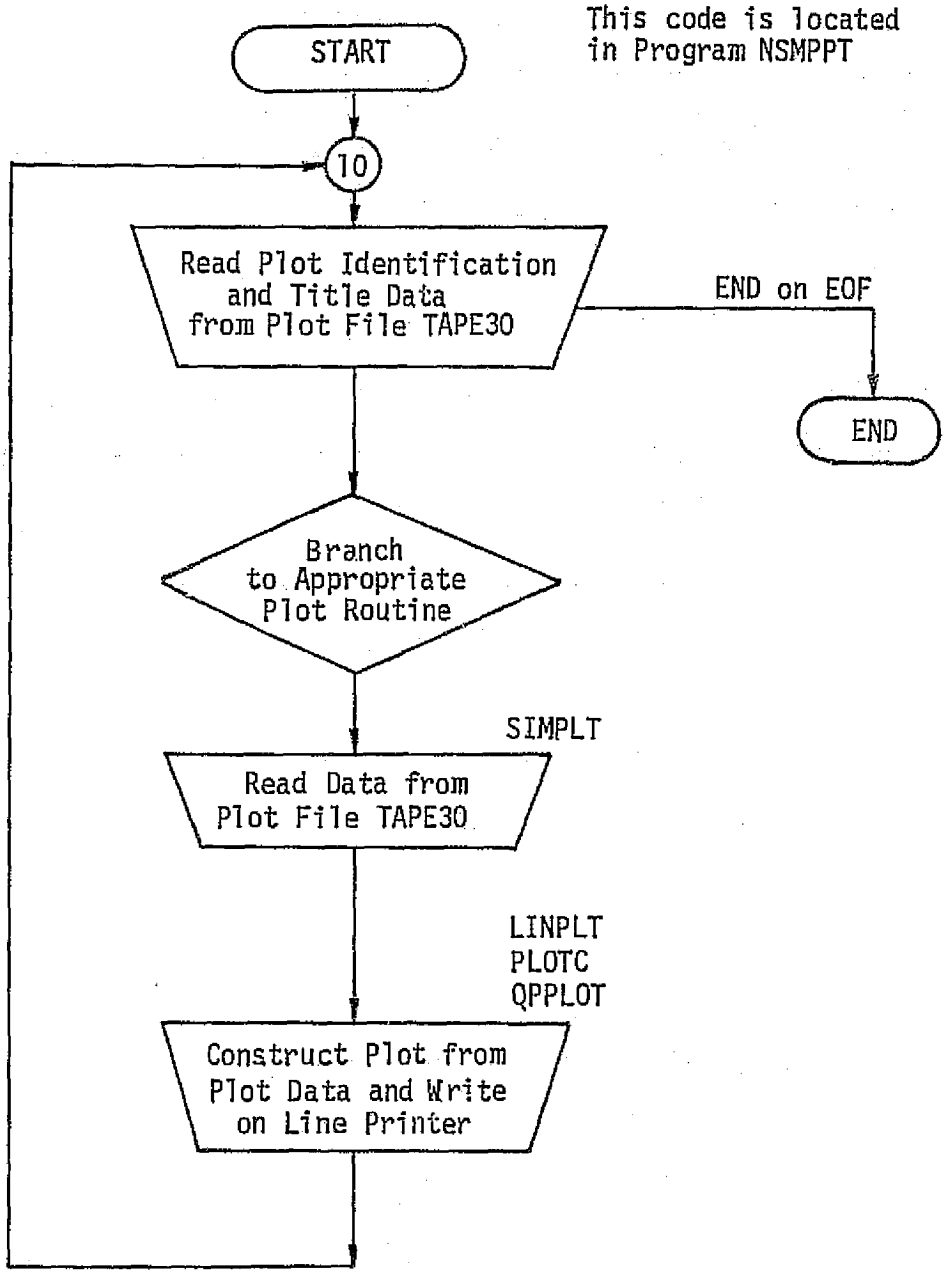

FIGURE 5.1-1 NSMPPT PROGRAM - MACRO FLOW DIAGRAM

```
CCENTER
      SUBROUTINE CENTER(ARRAY,NA,TITLE)
C
C     CENTERING TITLES
C
C
C     ARRAY CONTAINS CHARACTERS FOR TITLES OF PLOT
C     NA = NUMBER OF CHARACTERS IN ARRAY
C     TITLE = ARRAY CONTAINING CENTERED TITLE
C
      DIMENSION ARRAY(1),TITLE(1)
      DATA BLK/10H          /
      DO 10 I=1,12
      TITLE(I)=BLK
10    CONTINUE
      CALL NCHAR(ARRAY,NA,JSTART,NCH)
      IF (NCH.EQ.0) GO TO 200
      NN=(120-NCH)/2+1
      II=JSTART-1
      I=NN-1
      NEND=II+NCH
25    II=II+1
      I=I+1
      CALL GETT(ARRAY,II,AR)
      CALL PUTT(TITLE,I,AR)
      IF (II.LT.NEND) GO TO 25
200   RETURN
      END
```

```
CGNFPLT
      SUBROUTINE GNFPLT (W,I,J,K)
      READ (30) DUMMY
      IF ( EOF(30) ) 10,20
   10 K = 1
   20 RETURN
      END
```

```
CGRIDLI
      SUBROUTINE GRIDLI (NDVMAX,AMIN,AMAX,SMIN,SMAX,NDIV,NSIG)
C
C     PURPOSE - TO SELECT AXIS SCALES FOR A LINEAR AXIS.
C
C         AMIN,AMAX - MIN AND MAX VALUES OF THE DATA.
C         SMIN,SMAX - MIN AND MAX OF AXIS SCALES.
C              NDIV - NUMBER OF GRID DIVISIONS.
C              NSIG - NUMBER OF SIGNIFICANT FIGURES FOR ANNOTATION.
C
      IN = 1
      IF ( SMIN .NE. 0.0 .OR. SMAX .NE. 0.0 ) GO TO 40
C
C     SET AXIS INCREMENT TO 1,2 OR 5 * 10**N.
C
      IF ( ABS(AMIN-AMAX) .LE. 1.E-6*AMAX ) AMAX = 1.000001*AMIN
      IF ( AMAX .EQ. 0.0 .AND. AMIN .EQ. 0.0 ) AMAX = 1.E-6
      SPAN = ABS(AMAX-AMIN)
      STEP = SPAN / FLOAT(NDVMAX)
      N = ALOG10(STEP)
      IF ( STEP .LT. 1.0 ) N = N - 1
      X = STEP / 10.0**N
      IF ( X .GT. 2.0 ) GO TO 10
      STEP = 2.0 * 10.0**N
      GO TO 30
   10 IF ( X .GT. 5.0 ) GO TO 20
      STEP = 5.0 * 10.0**N
      GO TO 30
   20 STEP = 10.0**(N+1)
      IN = 0
   30 CONTINUE
C
C     SET SCALE MAX AND MIN.
C
      HSTEP = STEP * 0.5
      SMIN = AINT(AMIN/HSTEP) * HSTEP
      IF ( AMIN .LT. 0.0 ) SMIN = SMIN - HSTEP
      SMAX = AINT(AMAX/HSTEP) * HSTEP
      IF ( AMAX .GT. 0.0 ) SMAX = SMAX + HSTEP
      X = AMOD(ABS(SMIN),STEP)
      IF ( X .GT. 0.001*STEP .AND. X .LT. 0.999*STEP )
     +     SMIN = SMIN - HSTEP
      X = AMOD( SMAX-SMIN , STEP )
      IF ( X .GT. 0.001*STEP .AND. X .LT. 0.999*STEP )
     +     SMAX = SMAX + HSTEP
C
C     FIND NUMBER OF SUB-DIVISIONS.
C
      NDIV = (SMAX-SMIN) / STEP + 0.5
      GO TO 50
C
C     FIND NUMBER OF SIGNIFICANT FIGURES.
C
   40 CONTINUE
      STEP = ( SMAX - SMIN ) / FLOAT(NDIV)
```

```
50 CONTINUE
   XMAX = AMAX1(ABS(SMIN),ABS(SMAX))
   NMAX = ALOG10(XMAX*1.0001)
   NSTEP = ALOG10(STEP) * 1.00001
   IF ( STEP .LE. 1.0 .AND. XMAX .GE. 1.0 ) NSTEP = NSTEP - 1
   IF ( STEP .GE. 10.0 ) IN = 1
   NSIG = NMAX - NSTEP + IN
   RETURN
   END
```

```
CLEFTT
      SUBROUTINE LEFTT(ARRAY,LA)
C
C     LEFT TITLE
C
C
C     ARRAY CONTAINS CHARACTERS FOR LEFT TITLE
C     LA = NUMBER OF CHARACTERS IN ARRAY
C
      DIMENSION ARRAY(1)
      COMMON/CLEFTT/LEFT(51)
      COMMON/UNIT/IOUTP
      DATA BLK/4H    /,NN/51/
C
C     BLANK OUT LEFT ARRAY
C
      DO 2 I=1,NN
      LEFT(I)=BLK
2     CONTINUE
      IF (LA.EQ.0) RETURN
C
C     CENTER TITLE IN LEFT ARRAY
C
      CALL NCHAR(ARRAY,LA,ISTART,NCH)
      IF (NCH.LE.NN) GO TO 25
      NCH=NN
25    CONTINUE
      MUV=(NN-NCH)/2
      IEND=ISTART+NCH-1
      DO 30 I=ISTART,IEND
      MUV=MUV+1
      CALL GETT(ARRAY,I,LEFT(MUV))
30    CONTINUE
      RETURN
      END
```

```
CLINPLT
      SUBROUTINE LINPLT(X,Y,N,NTT,TT,NTL,TL,NTB1,TB1,NTB2,TB2,IAUTO)
C
C     SUBROUTINE TO DRAW PLOT VIA PLOTC
C
C
C     X = ARRAY OF POINTS FOR ABSCISSA
C     Y = ARRAY OF POINTS FOR ORDINATE
C     N = NUMBER OF POINTS TO BE PLOTTED
C     NTT = NUMBER OF CHARACTERS IN TOP TITLE
C     TT = ARRAY CONTAINING TOP TITLE
C     NTL = NUMBER OF CHARACTERS IN LEFT TITLE
C     TL = ARRAY CONTAINING LEFT TITLE
C     NTB1 = NUMBER OF CHARACTERS IN FIRST BOTTOM TITLE
C     TB1 = ARRAY CONTAINING FIRST BOTTOM TITLE
C     NTB2= NUMBER OF CHARACTERS IN BOTH SECOND AND THIRD BOTTOM TITLES
C     TB2 = ARRAY CONTAINING BOTH SECOND AND THIRD BOTTOM TITLES
C         TB2(I),I=1,20  CAN CONTAIN ONLY SECOND BOTTOM TITLE
C         TB2(I),I=21,40 CAN CONTAIN ONLY THIRD BOTTOM TITLE
C     IAUTO=0  AUTOMATIC SCALING
C     IAUTO=1  AXIS VALUES PROVIDED IN ZSCALE
C
      DIMENSION X(1),Y(1),TT(1),TL(1),TB1(1),TB2(1)
      DIMENSION TITLES(12)
      COMMON/UNIT/IOUTP
      IF (N.EQ.0) RETURN
C
C     CHECK FOR MULTIPLE CURVE PLOT
C
      IF (N.GT.0) GO TO 10
      CALL PLOTC(N,X,Y,IAUTO)
      RETURN
10    CONTINUE
C
C     DRAW PLOT WITH TITLES
C
      IND1=0
      IF (NTT.EQ.0) GO TO 15
      CALL CENTER(TT,NTT,TITLES)
      WRITE(IOUTP,101) TITLES
15    IF (NTT.EQ.0) WRITE(IOUTP,50)
      CALL LEFTT(TL,NTL)
      CALL PLOTC(N,X,Y,IAUTO)
      IF (NTB1.EQ.0) GO TO 20
      CALL CENTER(TB1,NTB1,TITLES)
      WRITE(IOUTP,100) TITLES
20    CONTINUE
      IF (NTB2.EQ.0) GO TO 40
      NDUM=NTB2
      IF (NTB2 .LE. 80) GO TO 30
      IND1=1
      N2=NTB2-80
      NDUM=80
30    CALL CENTER(TB2,NDUM,TITLES)
      WRITE(IOUTP,100) TITLES
      IF (IND1.NE.1) GO TO 40
```

```
      CALL CENTER(T82( 9),N2,TITLES)
      WRITE(IOUTP,100) TITLES
40    CONTINUE
      RETURN
50    FORMAT(1H1)
100   FORMAT(1H ,6X,12A10)
101   FORMAT(1H1,6X,12A10)
      END
```

```
CMNMX
      SUBROUTINE MNMX (A,N,AMIN,AMAX)
C
C     PURPOSE - TO FIND THE MINIMUM AND MAXIMUM VALUES OF AN ARRAY.
C
C                  A - ARRAY OF VALUES.
C                  N - NUMBER OF ELEMENTS IN A.
C          AMIN,AMAX - MIN AND MAX VALUES FOUND,  IF AMIN ¬= AMAX, THEN
C                      START WITH THE VALUES PASSED IN.
C
      DIMENSION A(1)
C
C     CHECK FOR ORIGINAL VALUES OF MIN AND MAX.
C
      IF ( AMIN .GT. AMAX ) GO TO 10
      IN = 1
      IF ( N .LT. 1 ) RETURN
      GO TO 20
C
C     INITIALIZE MIN AND MAX.
C
   10 CONTINUE
      AMIN = A(1)
      AMAX = A(1)
      IN = 2
      IF ( N .LT. 2 ) RETURN
C
C     SEARCH.
C
   20 CONTINUE
      DO 30 I=IN,N
      IF ( AMIN .GT. A(I) ) AMIN = A(I)
      IF ( AMAX .LT. A(I) ) AMAX = A(I)
   30 CONTINUE
      RETURN
      END
```

```
CNCHAR
      SUBROUTINE NCHAR(ARRAY,MAX,ISTART,NCH)
C
C     SUBROUTINE TO CALCULATE THE NUMBER OF CHARACTERS IN A CHARACTER
C         STRING
C
C
C     ARRAY CONTAINS CHARACTER STRING
C     NA = NUMBER OF INPUT CHARACTERS
C     ISTART = NUMBER OF FIRST NONBLANK CHARACTER IN STRING
C     NCH = NUMBER OF CHARACTERS IN ARRAY SUPPRESSING BEGINNING
C         AND ENDING BLANKS
C
      DIMENSION ARRAY(1)
      COMMON/UNIT/IOUTP
      DATA BLK/10H          /
      NCH=0
      J=0
5     J=J+1
      CALL GETT(ARRAY,J,AR)
      IF (AR.NE.BLK) GO TO 10
      IF (J.GE.MAX) GO TO 100
      GO TO 5
10    ISTART=J
      J=MAX+1
15    J=J-1
      CALL GETT(ARRAY,J,AR)
      IF (AR.NE.BLK) GO TO 20
      IF (J.LE.0) GO TO 100
      GO TO 15
20    NCH=J-ISTART+1
      IF (NCH.GE.0) GO TO 25
      NCH=0
      GO TO 100
25    CONTINUE
      IF (NCH.GT.120) NCH=120
100   CONTINUE
      RETURN
      END
```

```
CNSMPPT
      PROGRAM NSMPPT (OUTPUT,TAPE6=OUTPUT,TAPE30)
C
C             NONSIM OFFLINE PLOT PACKAGE.
C
      COMMON /CPLOTS/ IOPT(30),PLOTID( 5),PTITLE( 8)
      COMMON /CWORK/ WORK(3131)
      COMMON /UNIT/ IOUTP
      IOUTP = 6
      IERCNT = 0
      IEND = 0
C
C     READ THE OPTION AND TITLE ARRAYS.
C
   10 CONTINUE
      READ (30) IOPT,PLOTID,PTITLE
      IF ( EOF(30) ) 500,12
   12 CONTINUE
C
C     GENERAL PLOTS
C
      IF ( IOPT(1) .NE. 1 ) GO TO 30
      CALL GNFPLT (WORK,50,2,IEND)
      IF ( IEND .NE. 0 ) GO TO 500
      GO TO 10
C
C     SIMULATION PLOTS.
C
   30 CONTINUE
      IF ( IOPT(1) .NE. 2 ) GO TO 40
      CALL SIMPLT (WORK,IEND)
      IF ( IEND .NE. 0 ) GO TO 500
      GO TO 10
C
C     ROOT LOCUS PLOTS.
C
   40 CONTINUE
      IF ( IOPT(1) .NE. 3 ) GO TO 50
      CALL RTLPLT (WORK,IEND)
      IF ( IEND .NE. 0 ) GO TO 500
      GO TO 10
C
C     TRANSFER FUNCTION PLOTS.
C
   50 CONTINUE
      IF ( IOPT(1) .NE. 4 ) GO TO 60
      CALL TNFPLT (WORK,WORK(1001),WORK(2001),IEND)
      IF ( IEND .NE. 0 ) GO TO 500
      GO TO 10
C
C     STEADY STATE PLOTS.
C
   60 CONTINUE
      IF ( IOPT(1) .NE. 5 ) GO TO 400
      CALL SIMPLT (WORK,IEND)
```

```
      IF ( IEND .NE. 0 ) GO TO 500
      GO TO 10
C
C     ERROR
C
  400 CONTINUE
      IF ( IERCNT .GT. 10 ) GO TO 500
      WRITE (6,410)
  410 FORMAT (///1X,20(1H*),86H INCORRECT INTERMEDIATE PLOT DATA HAS BEE
     +N DETECTED.  CONTINUATION WILL BE ATTEMPTED. ,20(1H*)///)
      IERCNT = IERCNT + 1
      GO TO 10
C
C     EXIT.
C
  500 CONTINUE
      STOP
      END
```

```
CPLOTC
      SUBROUTINE PLOTC(M,X,Y,IAUTO)
C
C     SUBROUTINE WHICH CALLS PLOTTING SUBROUTINE QPPLOT
C
C
C     M = NUMBER OF POINTS TO BE PLOTTED
C     X = ARRAY OF POINTS FOR ABSCISSA
C     Y = ARRAY OF POINTS FOR ORDINATE
C     IAUTO=0  AUTOMATIC SCALING
C     IAUTO=1  AXIS VALUES PROVIDED IN ZSCALE
C
      COMMON/ZSCALE/XMAX,XMIN,YMAX,YMIN
      DIMENSION X(1),Y(1)
      DATA NUM /0/
      N=M
      NUM=NUM+1
      L=IABS(M)
      IF(M.GT.0) NUM =0
      IF(NUM.LT.8) GO TO 10
      NUM =0
      N=L
10    CONTINUE
      IF (IAUTO.EQ.0) CALL QXMXMN(X,Y,L,XMAX,XMIN,YMAX,YMIN)
      CALL QPPLOT(X,XMAX,XMIN,Y,YMAX,YMIN,N)
      RETURN
      END
```

```
CQPPLOT
      SUBROUTINE QPPLOT(TD,TMAX,TMIN,XD,XMAX,XMIN,NUMO)
C
C     PLOTTING SUBROUTINE
C
C
C     TD = ARRAY OF POINTS FOR ABSCISSA (Y-AXIS)
C     TMAX = MAXIMUM VALUE FOR TD-ARRAY
C     TMIN = MINIMUM VALUE FOR TD-ARRAY
C     XD = ARRAY OF POINTS FOR ORDINATE (X-AXIS)
C     XMAX = MAXIMUM VALUE FOR XD-ARRAY
C     XMIN = MINIMUM VALUE FOR XD-ARRAY
C     NUMO = NUMBER OF POINTS TO BE PLOTTED
C
C
C     GRAPHR IS A REAL*4 ARRAY CONTAINING PLOT - DIMENSION = 32X52
C     GRAPH IS A LOGICAL*1 ARRAY OF DIMENSION 128X52 WHICH IS
C         EQUIVALENCED TO GRAPHR
C     GRAPHR(1,I),I=1,51 CONTAINS  VERTICAL AXIS
C     GRAPHR(I,52),I=8,32 CONTAINS  HORIZONTAL AXIS
C     REST OF GRAPHR CONTAINS BORDERS AND ACTUAL PLOT
C
      COMMON/UNIT/IOUTP
      COMMON/CLEFTT/LEFT(51)
      DIMENSION GRAPHR(14,51),HAXIS(25),SCALE(10),POINT(12)
      DIMENSION FMTR(3),FMTD(6),FMTS(4),FMTB(8),VAL(2)
      DIMENSION TD(1),XD(1),TX(2,2),TXO(2),RANGE(2),DIV(2),DELT(2)
      DATA SCALE /1.,1.5,2.,3.,4.,5.,6.,8.,10.,15. /
      DATA K /0/
      DATA POINT /1H*,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9,1H ,1HX/
      DATA BLEFT,BRIGHT,HORIZ /10H+---      ,10H        ---,
     +                         10H----------/
      DATA NUMBER /10H5432100000/
      DATA BLANK /10H          /
C
C     OUTPUT FORMATS
C
      DATA FMTD /60H(1X,A1,1X,1PE9.2,12A10,A1)     (1X,A1,1X,F9.1,12A10,A
     +1)       /
      DATA FMTB /80H(4X,1P12E10.2,1PE9.2/9X,1P12E10.2)      (4X,0P12F10.
     +1,0PF9.1/9X,0P12F10.1)       /
      NUM=IABS(NUMO)
C
C     K IS NONZERO IMPLIES MULTIPLE CURVE PLOTS
C
      IF(K.NE.0)GO TO 1000
   50 CONTINUE
      IWT=0
      IP=0
      IF(NUMO.LT.0)IP=1
      TX(1,1)=TMIN
      TX(1,2)=TMAX
      TX(2,1)=XMIN
      TX(2,2)=XMAX
      VAL(2)=1.0
      VAL(1)=1.2
      ROUND=.9999
```

```
C
C       DETERMINE EVEN SCALES
        DO 600 L=1,2
C
        RANGE(L)=ABS(TX(L,2)-TX(L,1))
C
C       CHECK FOR CONSTANT VALUE
C
        IF(RANGE(L).EQ.0.)RANGE(L)=2.*ABS(TX(L,1))
        IF(RANGE(L).EQ.0.)RANGE(L)=10.
        N=ALOG10(RANGE(L))
        IF(RANGE(L).LT.1.)N=N-1
        EXPN=VAL(L)*10.**N
        DO 100 M=1,9
        K=M
        IF(SCALE(M)*EXPN.GE.RANGE(L)*ROUND)GO TO 150
  100   CONTINUE
  150   CONTINUE
        RANGE(L)=SCALE(K)*EXPN
        DIV(L)=RANGE(L)/10./VAL(L)
        TXMAX=AMAX1(ABS(TX(L,1)),ABS(TX(L,2)))*ROUND
        IF(TX(L,2)*TX(L,1).GE.0.)GO TO 300
C
C       TRY TO CENTER SCALE ABOUT ORIGIN
C
        IF(RANGE(L)/2..LT.TXMAX)GO TO 500
        TX0(L)=-RANGE(L)/2.
        GO TO 600
  300   CONTINUE
C
C       TRY TO START OR END SCALE AT ORIGIN
C
        IF(RANGE(L).LT.TXMAX)GO TO 500
        TX0(L)=0.
        IF(TX(L,1).LT.0.)TX0(L)=-RANGE(L)
        GO TO 600
  500   CONTINUE
C
C       FIND ORIGIN OF SCALE
C
        TX0(L)=TX(L,1)-AMOD(TX(L,1),DIV(L))
        IF(TX0(L).GT.TX(L,1))TX0(L)=TX0(L)-DIV(L)
C
C       INSURE THAT ALL POINTS FALL WITHIN SCALE RANGE
C
        IF(TX0(L)+RANGE(L).GE.TX(L,2)*ROUND)GO TO 600
        K=K+1
        GO TO 150
  600   CONTINUE
C
C       BLANK OUT PAGE
C
        DO 620 I=2,13
        DO 620 J=1,51
        GRAPHR(I,J)=BLANK
```

```
  620 CONTINUE
      DELT(1)=DIV(1)/10.
      DELT(2)=DIV(2)/5.
C
C     DEFINE VERTICAL AXIS AND BORDERS
C
      ICENTR=0
      IO=IFIX(1.5-TXO(1)/DELT(1))
      IF(IO.LT.1.OR.IO.GT.121)ICENTR=1
      IF(ICENTR.EQ.1)IO=1
      IZERO=0
      IF(ICENTR.NE.1)IZERO=IO/10
      XS=TXO(2)+RANGE(2)+DELT(2)
      DO 650 J=1,51
      CALL PUTT (GRAPHR(2,J),1,1H )
      CALL PUTT (GRAPHR(2,J),121,1H )
  650 CONTINUE
      IF ( IO .LE. 1 .OR. IO .GE. 121 ) GO TO 670
      DO 660 J=1,51
      CALL PUTT (GRAPHR(2,J),IO,1H.)
  660 CONTINUE
  670 CONTINUE
      DO 700 J=1,51,5
      GRAPHR(1,J)=XS-J*DELT(2)
      CALL STRMOV (BLEFT,1,4,GRAPHR(2,J),1)
      CALL PUTT (GRAPHR(2,J),IO,1H+)
      GRAPHR(13,J)=BRIGHT
      CALL PUTT (GRAPHR(2,J),121,1H+)
  700 CONTINUE
C
C     DEFINE HORIZONTAL AXIS AND BORDERS
C
      JO=IFIX(51.5+TXO(2)/DELT(2))
      IF(JO.LT.1.OR.JO.GT.51)ICENTR=2
      IF(ICENTR.EQ.2)JO=51
      J=0
      DIV(1)=DIV(1)/2.
      TS=TXO(1)
      HAXIS(1) = TS
      DO 750 I=2,13
      J=J+1
      HAXIS(I+12) = TS + J * DIV(1)
      J=J+1
      HAXIS(I) = TS + J * DIV(1)
  750 CONTINUE
C
C     AVOID ROUNDOFF IN CALCULATING ZERO POINT OF SCALES
C
      IF(IZERO.GT.0)HAXIS(IZERO) = 0.
      IF(ICENTR.NE.2)GRAPHR(1,JO)=0.
      DO 850 I=2,13
      GRAPHR(I,1)=HORIZ
      GRAPHR(I,JO)=HORIZ
      GRAPHR(I,51)=HORIZ
  850 CONTINUE
      DO 900 I=1,121,5
      CALL PUTT (GRAPHR(2,1),I,1H+)
```

```
      CALL PUTT (GRAPHR(2,J0),I,1H+)
      CALL PUTT (GRAPHR(2,51),I,1H+)
  900 CONTINUE
      DO 910 I=11,111,10
      CALL PUTT (GRAPHR(2,2),I,1H.)
      CALL PUTT (GRAPHR(2,3),I,1H.)
      CALL PUTT (GRAPHR(2,49),I,1H.)
      CALL PUTT (GRAPHR(2,50),I,1H.)
  910 CONTINUE
      IF ( ICENTR .EQ. 0 ) CALL PUTT (GRAPHR(2,J0),I0,1H0)
C
C     DEFINE FORMAT STATEMENT ACCORDING TO NUMERICAL RANGE OF DATA
C
      TXMAX=AMAX1(ABS(GRAPHR(1,1)),ABS(GRAPHR(1,51)))
      NS=ALOG10(TXMAX)+3.0001
      J=3
C
C     WILL AN *E* FORMAT BE REQUIRED FOR THE VERTICAL AXIS
C
      IF (NS.LT.1.OR.NS.GT.8) J=0
      DO 920 I=1,3
      FMTR(I)=FMTD(I+J)
  920 CONTINUE
      IF(J.EQ.0)GO TO 950
      NR=ALOG10(RANGE(2))+3.0001
      IF(NR.GT.NS)NR=NS
C
C     INSURE THAT THE FIELD CAN CONTAIN THE LARGEST NUMBER
C
      NS=MAX0(1,NR,NS-2)
      CALL GETT (NUMBER,NS,IJ)
      CALL PUTT (FMTR,14,IJ)
  950 CONTINUE
      TXMAX = AMAX1 (ABS(HAXIS(1)),ABS(HAXIS(25)))
      NS=ALOG10(TXMAX)+3.0001
      J = 4
C
C     WILL AN *E* FORMAT BE REQUIRED FOR THE HORIZONTAL AXIS
C
      IF (NS.LT.1.OR.NS.GT.8) J=0
      DO 970 I=1,4
      FMTS(I)=FMTB(I+J)
  970 CONTINUE
      IF (J.EQ.0) GO TO 1000
      NR=ALOG10(RANGE(1))+3.0001
      IF(NR.GT.NS)NR=NS
C
C     INSURE THAT THE FIELD CAN CONTAIN THE LARGEST NUMBER
C
      NS=MAX0(1,NR,NS-2)
      CALL GETT (NUMBER,NS,IJ)
      CALL PUTT (FMTS,13,IJ)
      CALL PUTT (FMTS,20,IJ)
      CALL PUTT (FMTS,33,IJ)
 1000 CONTINUE
      IP=IP+1
      IOFF=1
```

```
      IF(IP.GT.1)IOFF=IP-1
      M=0
      DO 1500 L=1,NUM
      LOC=IP
      I=IFIX(1.5+(TD(L)-TX0(1))/DELT(1))
      IF(I.LT.1.OR.I.GT.121)GO TO 1200
      J=IFIX(51.5-(XD(L)-TX0(2))/DELT(2))
      IF(J.LT.1.OR.J.GT.51)GO TO 1200
C
C     CHECK FOR MULTIPLE POINTS
C
      CALL GETT (GRAPHR(2,J),I,PGRAPH)
      IF ( PGRAPH .EQ. POINT(IP) ) GO TO 1500
      IF ( PGRAPH .EQ. POINT(1) ) GO TO 1500
C
C     THIS CHECK IS MACHINE DEPENDENT - CDC 6600
C
      IF ( PGRAPH .GT. POINT(2) .AND. PGRAPH .LE. POINT(10) ) LOC = 12
      CALL PUTT (GRAPHR(2,J),I,POINT(LOC))
      GO TO 1500
 1200 CONTINUE
      IWT=1
      M=M+1
 1500 CONTINUE
      IF(NUMO.LT.0.AND.IP.LT.10)GO TO 2000
      K=0
C
C     WRITE OUT PLOT
C
      DO 1700 I=1,51
      IF (MOD(I,5).EQ.1) GO TO 1600
      WRITE(IOUTP,1550) LEFT(I),(GRAPHR(J,I),J=2,14)
 1550 FORMAT (1X,A1,10X,12A10,A1)
      GO TO 1700
1600  WRITE(IOUTP,FMTR) LEFT(I),(GRAPHR(J,I),J=1,14)
1700  CONTINUE
      WRITE(IOUTP,FMTS) (HAXIS(J),J=1,25)
 2000 CONTINUE
      RETURN
      END
```

```
CQXMXMN
      SUBROUTINE QXMXMN(X,Y,N,AMAX,AMIN,OMAX,OMIN)
C
C     SUBROUTINE TO DETERMINE MINIMUM AND MAXIMUM VALUES OF ARRAYS
C
C
C     N = NUMBER OF PLOT POINTS
C     X = ARRAY OF POINTS FOR ABSCISSA
C     Y = ARRAY OF POINTS FOR ORDINATE
C     AMAX = MAXIMUM VALUE IN X-ARRAY
C     AMIN = MINIMUM VALUE IN X-ARRAY
C     OMAX = MAXIMUM VALUE IN Y-ARRAY
C     OMIN = MINIMUM VALUE IN Y-ARRAY
C
      DIMENSION X(1),Y(1)
      AMAX=-1.E50
      AMIN=1.E50
      OMAX=-1.E50
      OMIN=1.E50
      DO 1 I=1,N
      AMAX=AMAX1(X(I),AMAX)
      AMIN=AMIN1(X(I),AMIN)
      OMAX=AMAX1(Y(I),OMAX)
      OMIN=AMIN1(Y(I),OMIN)
    1 CONTINUE
      RETURN
      END
```

```
CRTLPLT
      SUBROUTINE RTLPLT (ROOT,IEND)
C
C     PURPOSE - TO BUILD A ROOT LOCUS PLOT FOR NONSIM.
C
C        ROOT - A WORK SPACE INTO WHICH DATA IS READ.
C
      COMMON /CPLOTS/ IOPT,ICASE,DATE(2),RLPAR,SCALR(3),SCALI(3),
     +                INDEX,DUMMY(18),PLOTID( 5),PTITLE( 8)
C
      COMMON /ZSCALE/ SMAXR,SMINR,SMAXI,SMINI
C
      DIMENSION ROOT(1)
      DIMENSION ZBFR(16)
      DIMENSION X(1000),Y(1000),GAIN(4,50),IGAIN(4,50)
      EQUIVALENCE (GAIN(1,1),IGAIN(1,1))
      DATA EPZ /1.0E-4/
C
C     READ ROOT ARRAY.
C
      READ (30) (ROOT(I),I=1,INDEX)
      IF ( EOF(30) ) 270,5
    5 CONTINUE
C
C     FIND MAX AND MIN VALUES.
C
      RMIN = 1.0
      RMAX = 0.0
      YMIN = 1.0
      YMAX = 0.0
      I = 1
   10 CONTINUE
      N = ROOT(I) + 0.1
      CALL MNMX (ROOT(I+3),N,RMIN,RMAX)
      CALL MNMX (ROOT(I+3+N),N,YMIN,YMAX)
      I = I + 2*N + 3
      IF ( I .LT. INDEX ) GO TO 10
      IF ( YMIN .LT. 0.0 ) YMIN = 0.0
C
C     FIND SCALE VALUES, IF THEY ARE NOT PROVIDED - REAL.
C
      IF ( SCALR(1) .LT. SCALR(2) ) GO TO 30
   20 CONTINUE
      SMINR = 0.0
      SMAXR = 0.0
      IAUTO = 0
      CALL GRIDLI (12,RMIN,RMAX,SMINR,SMAXR,NDIVR,NSIGR)
      GO TO 50
   30 CONTINUE
      SMINR = SCALR(1)
      SMAXR = SCALR(2)
      IAUTO = 1
   50 CONTINUE
```

```
C
C       FIND SCALE VALUES, IF THEY ARE NOT PROVIDED - IMAGINARY.
C
        IF ( SCALI(1) .LT. SCALI(2) ) GO TO 70
   60 CONTINUE
        SMINI = 0.0
        SMAXI = 0.0
        CALL GRIDLI (12,YMIN,YMAX,SMINI,SMAXI,NDIVI,NSIGI)
        GO TO 90
   70 CONTINUE
        SMINI = SCALI(1)
        SMAXI = SCALI(2)
   90 CONTINUE
        EPZR = (SMAXR-SMINR) * 0.002
        EPZI = (SMAXI-SMINI) * 0.002
C
C       DECOMPOSE ROOT ARRAY AND GUARANTEE SEPARATION OF ROOTS.
C
        X(1) = 1.E69
        Y(1) = 1.E69
        I = 1
        N = 0
        NR = 0
        INDRT = 0
  100 CONTINUE
        IF ( I .GT. INDEX ) GO TO 160
        NN = ROOT(I) + 0.1
        NR = NR + 1
        GAIN(1,NR) = ROOT(I+1)
        GAIN(2,NR) = ROOT(I+2)
        IGAIN(3,NR) = N + 1
        I1 = I + 2
        I2 = I1 + NN
        IF ( GAIN(2,NR) .EQ. 5.0 .AND. INDRT .EQ. 1 ) GO TO 120
        IF ( GAIN(2,NR) .EQ. 5.0 ) INDRT = 1
        DO 110 J=1,NN
        N = N + 1
        X(N) = ROOT(I1+J)
        Y(N) = ROOT(I2+J)
  110 CONTINUE
        IGAIN(4,NR) = NN
        GO TO 150
  120 CONTINUE
        NK = 0
        DO 140 J=1,NN
        RR = ROOT(I1+J)
        RI = ROOT(I2+J)
        IF ( ABS(RR) .LE. EPZ ) RR = 0.0
        IF ( ABS(RI) .LE. EPZ ) RI = 0.0
        DO 130 K=1,N
        IF ( ABS(RR-X(K)) .GT. EPZR   ) GO TO 130
        IF ( ABS(RI-Y(K)) .LE. EPZI   ) GO TO 140
  130 CONTINUE
        NK = NK + 1
        N = N + 1
        X(N) = RR
        Y(N) = RI
```

```
  140 CONTINUE
      IGAIN(4,NR) = NK
      IF ( NK .LE. 0 ) NR = NR - 1
  150 CONTINUE
      I = I + 3 + 2 * NN
      GO TO 100
C
C     GENERATE LABELS AND PLOT ON PRINTER
C
  160 CONTINUE
      ENCODE (80,250,ZBFR) RLPAR
  250 FORMAT (23HROOT LOCUS PARAMETER = ,A8,49X)
      ENCODE (80,260,ZBFR(9)) DATE,ICASE
  260 FORMAT (2A12,14X,15HROOT LOCUS PLOT,15X,8HCASE NO.,I4)
      CALL LINPLT (X,Y,N,80,PTITLE,10,10HIMAGINARY ,4,4HREAL,
     +             160,ZBFR,IAUTO)
C
C     ADVANCE FILM AND RETURN
C
      RETURN
  270 CONTINUE
      IEND = 1
      RETURN
      END
```

```
CSIMPLT
      SUBROUTINE SIMPLT (DSPLY,IEND)
C
C     PURPOSE - TO BUILT A SERIES OF SIMULATION (OR STEADY-STATE)
C               PLOTS, UP TO FIVE GRIDS PER PLOT.
C
      DIMENSION DSPLY(3131),VAR(31)
C
      COMMON /CPLOTS/ IOPT,ICASE,DATE(2),NPLT,NGRD(6),INDEX,NCODES,
     +                IMANUL,NWORK,DUMMY(15),PLOTID( 5),PTITLE(8)
C
      COMMON /ZSCALE/ XMAX,XMIN,YMAX,YMIN
C
      DIMENSION SCALE(5,4,6),PNAME(5,2,6),NPOS(5,2,6)
C
      DIMENSION ZBFR(8)
      DATA BLNK /10H          /
C
C     READ DATA.
C
      READ (30) SCALE,PNAME,NPOS
      IF ( EOF(30) ) 500,5
    5 CONTINUE
C
C     READ SIMULATION DATA.
C
      IL = 0
      INDMAX = 3131 / NCODES
  100 CONTINUE
      I1 = IL + 1
      IL = IL + INDMAX
      IF ( INDEX .LT. IL ) IL = INDEX
      INDX = IL - I1 + 1
      J = 0
      DO 130 I=I1,IL
      J = J + 1
      READ (30) VAR
      IF ( EOF(30) ) 500,110
  110 CONTINUE
      DO 120 K=1,NCODES
      DSPLY(INDMAX*(K-1)+J) = VAR(K)
  120 CONTINUE
  130 CONTINUE
C
C     INCREMENT OVER THE NUMBER OF PLOTS AND THE NUMBER OF GRIDS.
C
      DO 60 IP=1,NPLT
      NG = NGRD(IP)
      DO 40 IG=1,NG
C
C     SET SCALE VALUES IF REQUIRED.
C
      IAUTO = 0
      IF ( IMANUL .EQ. 1 ) GO TO 10
      GO TO 20
```

```
   10 CONTINUE
      IF ( SCALE(IG,1,IP) .GE. SCALE(IG,2,IP) .OR.
     +     SCALE(IG,3,IP) .GE. SCALE(IG,4,IP) ) GO TO 20
      IAUTO = 1
      XMAX = SCALE(IG,4,IP)
      XMIN = SCALE(IG,3,IP)
      YMAX = SCALE(IG,2,IP)
      YMIN = SCALE(IG,1,IP)
   20 CONTINUE
C
C     TITLES AT TOP OF PLOT.
C
      IF ( IOPT .EQ. 5 ) GO TO 52
      ENCODE (80,50,ZBFR) DATE,IP,ICASE
   50 FORMAT (2A12,12X,18HSIMULATION DISPLAY,I2,12X,8HCASE NO.,I4)
      GO TO 58
   52 CONTINUE
      ENCODE (80,54,ZBFR) DATE,IP,CASE
   54 FORMAT (2A12,11X,20HSTEADY STATE DISPLAY,I2,11X,8HCASE NO.,I4)
   58 CONTINUE
C
C     CALL PRINTER PLOTTER
C
      INX = INDMAX * (NPOS(IG,2,IP)-1) + 1
      INY = INDMAX * (NPOS(IG,1,IP)-1) + 1
      CALL LINPLT (DSPLY(INX),DSPLY(INY),INDX,80,PTITLE,
     +             8,PNAME(IG,1,IP),8,PNAME(IG,2,IP),80,ZBFR,IAUTO)
   40 CONTINUE
   60 CONTINUE
      IF ( IL+1 .LT. INDEX ) GO TO 100
      RETURN
  500 CONTINUE
      IEND = 1
      RETURN
      END
```

```
CTNFPLT
      SUBROUTINE TNFPLT (F,G,P,K)
      READ (30) DUMMY
      IF ( EOF(30) ) 10,20
   10 K = 1
   20 RETURN
      END
```